

Guía para desarrolladores de la versión 2.x

AWS SDK for Java 2.x



AWS SDK for Java 2.x: Guía para desarrolladores de la versión 2.x

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon, de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

Guía para desarrolladores. AWS SDK for Java 2.x	1
Introducción al SDK	1
Desarrollar aplicaciones móviles	1
Mantenimiento y compatibilidad de las versiones principales del SDK	1
Recursos adicionales	2
Contribución al SDK	2
Tutorial de introducción	3
Paso 1: Configuración para este tutorial	3
Paso 2: Crear el proyecto	3
Paso 3: Escribir el código	8
Paso 4: Compilar y ejecutar la aplicación	12
Success	13
Limpieza	13
Sigüientes pasos	13
Configuración	15
Descripción general de la configuración	15
Configuración de la autenticación	16
Configuración para el acceso mediante inicio de sesión único para el SDK	16
Inicie sesión con el AWS CLI	17
Instalar Java y una herramienta de compilación	18
Opciones de autenticación adicionales	18
Configurar un proyecto Apache Maven	18
Requisitos previos	19
Crear un proyecto de Maven	19
Configurar el compilador Java para Maven	20
Declarar el SDK como una dependencia	21
Establecer dependencias para módulos del SDK	22
Compilación del proyecto	24
Configurar un proyecto Gradle	25
Configurar un proyecto de imagen nativa de GraalVM	31
Requisitos previos	31
Crear un proyecto mediante el arquetipo	32
Crear una imagen nativa	33
Usar el SDK	34

Trabajar con los clientes de servicios	34
Crear un cliente de servicio	34
Configuración predeterminada de cliente	35
Configurar clientes de servicio	35
Hacer solicitudes	36
Tratamiento de respuestas	36
Cerrar el cliente de servicio	37
Tratamiento de excepciones	38
Utilizar esperadores	39
Clientes de HTTP	39
Tiempos de espera	40
Interceptores de ejecución	41
Información adicional	41
Proporcionar credenciales temporales al SDK	41
Configurar el acceso a las credenciales temporales	41
Cadena predeterminada de proveedores de credenciales	43
Usar un determinado proveedor de credenciales o una cadena de proveedores	45
Perfiles de usuario	46
Cargar las credenciales temporales de un proceso externo	49
Proporcione las credenciales temporales mediante código	52
Lea las credenciales del rol de IAM en Amazon EC2	55
Uso Regiones de AWS	57
Configure explícitamente una Región de AWS	57
Determinar la región desde el entorno	58
Comprobación de la disponibilidad del servicio	59
Seleccionar un punto de conexión específico	60
Reduzca el tiempo de inicio del SDK para AWS Lambda	60
Utilice un cliente HTTP AWS basado en CRT	60
Eliminar las dependencias del cliente HTTP no utilizadas	61
Configurar los clientes de servicio para abreviar las búsquedas	62
Inicializar el cliente de SDK fuera del controlador de la función de Lambda	63
Minimizar la inyección de dependencias	64
Usa un arquetipo de segmentación de Maven AWS Lambda	64
Lambda SnapStart	64
Cambios en la versión 2.x que afectan al tiempo de startup	64
Recursos adicionales de	64

Clientes de HTTP	65
Clientes disponibles	65
Recomendaciones sobre clientes	66
Valores predeterminados inteligentes	69
Configurar el cliente HTTP basado en Apache	71
Configurar el cliente HTTP basado en URLConnection	77
Configurar el cliente HTTP basado en Netty	83
Configurar clientes AWS HTTP basados en CRT	89
Configurar proxies HTTP	101
Tratamiento de excepciones	106
¿Por qué excepciones no controladas?	106
AwsServiceException (y subclases)	107
SdkClientException	108
Excepciones y comportamiento de reintentos	108
Reintentos	108
Estrategias de reintento	109
Especifique una estrategia	113
Personaliza una estrategia	114
Migración desde RetryPolicy a RetryStrategy	116
Registro	117
Archivo de configuración de Log4j 2	117
Añadir dependencia de registro	117
Errores y advertencias específicos del SDK	118
Registro de resumen de solicitud/respuesta	119
Registro del SDK a nivel de depuración	120
Habilite el registro de cables	122
Configurar el TTL de JVM para las búsquedas de nombres DNS	127
Cómo configurar el TTL de JVM	128
Prácticas recomendadas	129
Reutilizar un cliente del SDK	129
Cerrar las secuencias de entrada	129
Ajustar las configuraciones HTTP	130
Utilizar OpenSSL para Netty	130
Configurar los tiempos de espera de la API	130
Utilizar métricas	132
Resolución de problemas	132

Conexión restablecida	132
Tiempo de espera de la conexión	133
Se agotó el tiempo de espera de lectura	133
Tiempo de espera del grupo de conexiones	134
Errores de Classpath	138
¿Errores de firma	139
¿Se ha cerrado el grupo de conexiones	140
Uso de características del SDK	143
Características generales	143
Características específicas de servicios	143
Trabaje con resultados paginados	143
Paginación síncrona	144
Paginación asíncrona	147
Sondear los estados de recursos	152
Requisitos previos	153
Uso de esperadores	153
Configurar los esperadores	154
Ejemplos de código	155
Usar la programación asíncrona	155
Operaciones sin streaming	156
Operaciones de streaming	159
Operaciones avanzadas	162
Trabajar con HTTP/2	164
Usar las métricas del SDK	164
Requisitos previos	164
Habilitar la recopilación de métricas	165
Personalice el publicador de métricas	167
¿Cuándo están disponibles las métricas?	169
¿Qué información se recopila?	169
¿Cómo puedo usar esta información?	169
Métricas de los clientes de servicio	170
Trabaje con Servicios de AWS	175
CloudWatch	175
Obtener métricas de CloudWatch	176
Publicar datos de métricas personalizadas para CloudWatch	178
Uso de alarmas de CloudWatch	180

Usa Amazon CloudWatch Events	184
AWS servicios de bases de datos	188
Amazon DynamoDB	188
Amazon RDS	189
Amazon Redshift	189
Amazon Aurora sin servidor v1	190
Amazon DocumentDB	190
DynamoDB	190
Trabaja con tablas en DynamoDB	191
Uso de elementos en DynamoDB	201
Asignar objetos a elementos de DynamoDB	208
Amazon EC2	328
Administrar instancias Amazon EC2	328
Zonas de uso Regiones de AWS y disponibilidad	335
Trabaje con grupos de seguridad en Amazon EC2	338
Trabajar con metadatos de la instancia de Amazon EC2	343
IAM	349
Administrar las claves de IAM acceso	350
Administrar usuarios de IAM	356
Crear políticas de IAM	360
Trabajar con políticas de IAM	368
Trabajar con certificados IAM de servidor	374
Kinesis	379
Suscribirse a Amazon Kinesis Data Streams	380
Lambda	391
Invocar una función Lambda	391
Enumerar funciones de Lambda	392
Eliminación de una función de Lambda	393
Amazon S3	394
Uso de puntos de acceso o puntos de acceso de varias regiones	395
Operaciones con buckets	396
Operaciones con objetos	403
URL prefirmadas	413
Acceso entre regiones	422
Sumas de comprobación	423
Utilizar un cliente S3 de alto rendimiento	425

Transferir archivos y directorios	427
Amazon SNS	435
Crear un tema	435
Enumerar sus temas de Amazon SNS	436
Suscribir un punto de enlace a un tema	437
Publicar un mensaje en un tema	438
Cancelar la suscripción de un punto de enlace de un tema	439
Eliminación de un tema	440
Amazon SQS	441
Operaciones de cola	442
Operaciones con mensajes	445
Amazon Transcribe	449
Configurar el micrófono	449
Crear un editor	450
Cree el cliente e inicie la secuencia	452
Más información	448
Ejemplos de código	455
Acciones y escenarios	455
API Gateway	457
Aplicación de escalado automático	461
Controlador de recuperación de aplicaciones	470
Aurora	473
Auto Scaling	508
Amazon Bedrock	570
Amazon Bedrock Runtime	576
CloudFront	657
CloudWatch	677
CloudWatch Eventos	728
CloudWatch Registros	734
Amazon Cognito Identity	745
Amazon Cognito Identity Provider	752
Amazon Comprehend	780
DynamoDB	791
Amazon EC2	871
Amazon ECR	943
Amazon ECS	983

Elastic Load Balancing: versión 2	997
MediaStore	1042
OpenSearch Servicio	1057
EventBridge	1066
Previsión	1098
AWS Glue	1112
HealthImaging	1136
IAM	1164
AWS IoT	1249
AWS IoT data	1276
Amazon Keyspaces	1278
Kinesis	1305
AWS KMS	1318
Lambda	1355
MediaConvert	1384
Migration Hub	1407
Amazon Personalize	1420
Eventos de Amazon Personalize	1450
Versión ejecutable de Amazon Personalize	1454
Amazon Pinpoint	1458
API de SMS y voz de Amazon Pinpoint	1503
Amazon Polly	1507
Amazon RDS	1513
Amazon Redshift	1556
Amazon Rekognition	1582
Registro de dominios de Route 53	1650
Amazon S3	1673
S3 Glacier	1815
SageMaker	1831
Secrets Manager	1860
Amazon SES	1862
Amazon SES API v2	1875
Amazon SNS	1893
Amazon SQS	1960
Step Functions	1998
AWS STS	2022

AWS Support	2025
Systems Manager	2049
Amazon Textract	2079
Amazon Transcribe	2090
Ejemplos de servicios cruzados	2107
Creación de una aplicación para enviar datos a una tabla de DynamoDB	2107
Crear un chatbot de Amazon Lex	2108
Creación de una aplicación de Amazon SNS	2108
Crear una aplicación de mensajería	2109
Creación de una aplicación sin servidor para administrar fotos	2109
Creación de una aplicación web para hacer un seguimiento de los datos de DynamoDB ...	2110
Crear una aplicación web para realizar un seguimiento de los datos de Amazon Redshift .	2110
Crear un rastreador de elementos de trabajo de Aurora Serverless	2111
Creación de una aplicación para analizar los comentarios de los clientes	2111
Detección de EPI en imágenes	2112
Detectar objetos en imágenes	2112
Detecte personas y objetos en un video	2113
Supervisión del rendimiento de DynamoDB	2113
Uso de API Gateway para invocar una función de Lambda	2114
Usar Step Functions para invocar funciones de Lambda	2114
Usar eventos programados para invocar una función de Lambda	2115
Seguridad	2116
Protección de datos	2116
seguridad de la capa de transporte (TLS)	2118
Consultar las versiones de TLS	2118
Aplicar versiones de TLS	2119
Migrar a TLS 1.2	2119
Identity and Access Management	2119
Público	2120
Autenticación con identidades	2120
Administración de acceso mediante políticas	2124
¿Cómo Servicios de AWS trabajar con IAM	2127
Solución de problemas de AWS identidad y acceso	2127
Validación de la conformidad	2129
Resiliencia	2131
Seguridad de infraestructuras	2131

Migrar a la versión 2	2133
¿Qué novedades incluye la versión 2?	2133
tep-by-step Instrucciones S	2134
Información general sobre los pasos	2134
Ejemplo de migración	2136
Nombre del paquete a las asignaciones de ArtifactID	2147
Convención de nomenclatura	2147
Excepciones	2148
Diferencias entre 1.x y 2.x	2153
Cambio de nombre de paquete	2153
Adición de la versión 2.x a su proyecto	2154
POJO inmutables	2155
Métodos Setter y Getter	2155
Nombres de clases de modelos	2156
Bibliotecas y utilidades	2156
Cambios de cliente	2158
Cambios en el proveedor de credenciales	2202
Cambios de región	2210
Cambios en las operaciones, las solicitudes y las respuestas	2212
Cambios de excepción	2214
Cambios de serialización	2215
Cambios específicos de los servicios	2216
Cambios en el archivo de perfil	2222
Configuración externa	2223
Esperadores	2226
S3 Transfer Manager	2231
Utilidad de metadatos EC2	2237
CloudFront prefirmar	2245
Análisis de URI de S3	2249
API del creador de políticas de IAM	2251
Utilizar el SDK para Java 1.x y 2.x en paralelo	2257
Clave OpenPGP	2259
Clave actual	2259
Historial de documentos	2261
.....	mmcclxix

Guía para desarrolladores. AWS SDK for Java 2.x

AWS SDK for Java proporciona la API de Java para Servicios de AWS. Con el SDK, le resultará fácil crear aplicaciones Java que funcionen con Amazon S3, Amazon EC2, DynamoDB y otras.

El AWS SDK for Java 2.x es un cambio importante con respecto a la base de código de la versión 1.x. Se basa en Java 8 y agrega varias características solicitadas con frecuencia. Entre estas se incluyen la compatibilidad con operaciones de E/S sin bloqueo y la capacidad de conectar una implementación HTTP diferente en tiempo de ejecución.

Añadimos periódicamente nuevos servicios a AWS SDK for Java. Para obtener una lista de los cambios realizados y las características en una versión determinada, consulte el [registro de cambios](#).

Introducción al SDK

Si está listo para empezar a utilizar el SDK, consulte el capítulo [Tutorial de introducción](#).

Para configurar su entorno de desarrollo, consulte [Configuración](#).

Si actualmente utiliza la versión 1.x del SDK for Java, consulte [Migrar a la versión 2](#) para obtener instrucciones específicas.

Para obtener información sobre cómo hacer solicitudes a Amazon S3, DynamoDB, Amazon EC2 y otros Servicios de AWS, consulte [Usar el SDK for Java](#) y [Trabajar con Servicios de AWS](#).

Desarrollar aplicaciones móviles

Si es usted desarrollador de aplicaciones móviles, Amazon Web Services le proporciona el marco [AWS Amplify](#).

Mantenimiento y compatibilidad de las versiones principales del SDK

Para obtener información sobre el mantenimiento y la compatibilidad con las principales versiones del SDK y sus dependencias subyacentes, consulte los temas siguientes en la [Guía de Referencia de SDK y herramientas de AWS](#):

- [Política de mantenimiento de SDK y herramientas de AWS](#)
- [Matriz de compatibilidad para versiones de SDK y herramientas de AWS](#)

Recursos adicionales

Además de esta guía, estos son algunos otros recursos online útiles para los desarrolladores de AWS SDK for Java:

- [Referencia de la API del AWS SDK for Java 2.x](#)
- [Blog para desarrolladores de Java](#)
- [Tema de desarrollo de Java en AWS re:Post](#)
- [Código fuente del SDK en GitHub](#)
- [Biblioteca de ejemplos de códigos del SDK de AWS](#)
- [@awsforjava \(Twitter\)](#)

Contribución al SDK

Los desarrolladores también pueden enviar sus comentarios a través de los siguientes canales:

- Enviar problemas en GitHub:
 - [Enviar los problemas de documentación de la Guía para desarrolladores](#)
 - [Enviar problemas del SDK](#)
- Unirse a un chat informal sobre el SDK en el [canal de gitter](#) del AWS SDK for Java 2.x.

Comience con la versión AWS SDK for Java 2.x

AWS SDK for Java 2.x Proporciona las API de Java para Amazon Web Services (AWS). Con el SDK, puede crear aplicaciones Java que funcionen con Amazon S3 Amazon EC2, DynamoDB, y más.

En este tutorial, se muestra cómo usar [Apache Maven](#) para definir las dependencias del SDK para Java 2.x y, a continuación, escribir el código al que se conecta Amazon S3 para cargar un archivo.

Para completar el tutorial, siga estos pasos:

- [Paso 1: Configuración para este tutorial](#)
- [Paso 2: Crear el proyecto](#)
- [Paso 3: Escribir el código](#)
- [Paso 4: Compilar y ejecutar la aplicación](#)

Paso 1: Configuración para este tutorial

Antes de empezar este tutorial, necesitará:

- Permiso de acceso Amazon S3
- Un entorno de desarrollo de Java que está configurado para acceder Servicios de AWS mediante un inicio de sesión único al AWS IAM Identity Center

Use las instrucciones de [???](#) para configurar para este tutorial. Una vez que haya [configurado su entorno de desarrollo con acceso de inicio de sesión único para el SDK de Java y haya tenido una sesión activa en el portal de AWS acceso](#), continúe con el paso 2 de este tutorial.

Paso 2: Crear el proyecto

Para crear el proyecto de este tutorial, ejecute un comando de Maven que le pida información sobre cómo configurar el proyecto. Una vez introducidas y confirmadas todas las entradas, Maven termina de construir el proyecto creando un archivo `pom.xml` y archivos `stub` de Java.

1. Abra una ventana de terminal o línea de comandos y acceda a un directorio de su elección, como su carpeta Desktop o Home.

2. Introduzca el siguiente comando en el terminal y pulse Enter.

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

3. Introduzca el valor que aparece en la segunda columna para cada pregunta.

Prompt	Valor para introducir
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. Después de introducir el último valor, Maven muestra una lista de sus elecciones. Confirme introduciendo *Y* o vuelva a introducir los valores respondiendo con *N*

Maven crea la carpeta del proyecto llamada `getstarted` basándose en el valor `artifactId` que haya introducido. En la carpeta `getstarted`, busque un archivo `README.md` que pueda revisar, un archivo `pom.xml` y un directorio `src`.

Maven crea el árbol de directorios siguiente.

```
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
```

A continuación se muestra el contenido del archivo de proyecto `pom.xml`.

pom.xml

La sección `dependencyManagement` contiene una dependencia con el AWS SDK for Java 2.x y la sección `dependencies` una dependencia para Amazon S3. El proyecto usa Java 1.8 debido al valor 1.8 de las propiedades `maven.compiler.target` y `maven.compiler.source`.

```
<?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>
```

Paso 3: Escribir el código

En el código siguiente se muestra la clase `App` creada por Maven. El método `main` es el punto de entrada a la aplicación, que crea una instancia de la `Handler` clase y, a continuación, llama a su método `sendRequest`.

Clase `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");
    }
}
```

La clase `DependencyFactory` creada por Maven contiene el método de fábrica `s3Client` que crea y devuelve una instancia [S3Client](#). La instancia `S3Client` usa una instancia del cliente HTTP basado en Apache. Esto se debe a que especificó `apache-client` cuando Maven preguntó qué cliente HTTP usar.

La `DependencyFactory` muestra en el siguiente código.

Clase `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();
    }
}
```

La clase `Handler` contiene la lógica principal del programa. Cuando se crea una instancia de `Handler` en la clase `App`, `DependencyFactory` proporciona el cliente de servicios `S3Client`. El código utiliza la instancia de `S3Client` para llamar al servicio Amazon S3.

Maven genera la siguiente clase `Handler` con un comentario `TODO`. El siguiente paso del tutorial reemplaza el `TODO` por código.

Clase **Handler**, generada por 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 completar la lógica, reemplace todo el contenido de la clase `Handler` por el código siguiente. El método `sendRequest` se rellena y se añaden las importaciones necesarias.

Clase **Handler**, implementada

En primer lugar, el código crea un nuevo bucket de S3 con la última parte del nombre generado utilizando `System.currentTimeMillis()` para que el nombre del bucket sea único.

Tras crear el bucket en el método `createBucket()`, el programa carga un objeto mediante el método [putObject](#) de `S3Client`. El contenido del objeto es una cadena simple creada con el método `RequestBody.fromString`.

Por último, el programa elimina el objeto seguido del bucket del 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");
}
}
```

Paso 4: Compilar y ejecutar la aplicación

Una vez creado el proyecto y que contenga la clase `Handler` completa, compile y ejecute la aplicación.

1. Asegúrese de que tenga una sesión activa del Centro de identidades de IAM. Para ello, ejecute el comando `aws sts get-caller-identity` de la AWS Command Line Interface y compruebe la respuesta. Si no tiene una sesión activa, consulte [esta sección](#) para obtener instrucciones.
2. Abra una ventana de terminal o símbolo del sistema y desplácese hasta el directorio del proyecto `getstarted`.
3. Para compilar su proyecto, utilice el comando siguiente:

```
mvn clean package
```

4. Para ejecutar la aplicación, utilice el comando siguiente.

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

Para ver el bucket y el objeto nuevos creados por el programa, siga los pasos descritos a continuación.

1. En `Handler.java`, comente la línea `cleanup(s3Client, bucket, key)` del método `sendRequest` y guarde el archivo.
2. Recompile el proyecto ejecutando `mvn clean package`.
3. Vuelva a ejecutar `mvn exec:java -Dexec.mainClass="org.example.App"` para cargar el objeto de texto una vez más.
4. Inicie sesión en [la consola de S3](#) para ver el nuevo objeto en el bucket recién creado.

Después de ver el archivo, elimine el objeto y, a continuación, elimine el bucket.

Success

Si su proyecto de Maven se creó y ejecutó sin errores, ¡enhorabuena! Ha creado correctamente su primera aplicación Java mediante SDK para Java 2.x.

Limpieza

Para limpiar los recursos que ha creado en este tutorial, haga lo siguiente:

- Si aún no lo ha hecho, en [la consola de S3](#), elimine todos los objetos y buckets creados al ejecutar la aplicación.
- Elimine la carpeta del proyecto (`getstarted`).

Siguientes pasos

Ahora que ya ha aprendido lo básico, puede aprender lo siguiente:

- [Trabajando con Amazon S3](#)

- [Trabajar con otros Amazon Web Services DynamoDB Amazon EC2, como y varios servicios de bases de datos](#)
- [Usar el SDK](#)
- [Seguridad para el AWS SDK for Java](#)

Configura el AWS SDK for Java 2.x

En esta sección se ofrece información sobre cómo configurar su entorno de desarrollo y proyectos para utilizar el AWS SDK for Java 2.x.

Descripción general de la configuración

Para desarrollar correctamente aplicaciones a las que se acceda Servicios de AWS mediante el AWS SDK for Java, se requieren las siguientes condiciones:

- El SDK de Java debe tener acceso a las credenciales para [autenticar las solicitudes](#) en su nombre.
- Los [permisos de la función de IAM](#) configurada para el SDK deben permitir el acceso a los Servicios de AWS que requiera la aplicación. Los permisos asociados a la política PowerUserAccess AWS gestionada son suficientes para la mayoría de las necesidades de desarrollo.
- Un entorno de desarrollo con los siguientes elementos:
 - [Archivos de configuración compartidos](#) que se configuran al menos de una de las siguientes maneras:
 - El config archivo contiene la [configuración de inicio de sesión único del IAM Identity Center](#) para que el SDK pueda obtener las credenciales. AWS
 - El archivo `credentials` contiene credenciales temporales.
 - Una [instalación de Java 8](#) o posterior.
 - Una [herramienta de automatización de compilaciones](#), como [Maven](#) o [Gradle](#).
 - Un editor de texto para trabajar con código.
 - (Opcional, pero recomendado) Un IDE (entorno de desarrollo integrado) como [IntelliJ IDEA](#), Eclipse o [NetBeans](#)

Cuando utilizas un IDE, también puedes integrarlo AWS Toolkit para trabajar con él más fácilmente. Servicios de AWS El [AWS Toolkit para IntelliJ](#) y el [AWS Toolkit for Eclipse](#) son dos kits de herramientas que puede utilizar para el desarrollo de Java.

- Una sesión activa en el portal de AWS acceso cuando esté listo para ejecutar la aplicación. Se utiliza AWS Command Line Interface para [iniciar el proceso de inicio de sesión en el](#) portal de AWS acceso del Centro de Identidad de IAM.

Important

En las instrucciones de esta sección de configuración se supone que usted o su organización utilizan el Centro de identidad de IAM. Si su organización utiliza un proveedor de identidad externo que funciona de forma independiente del Centro de identidades de IAM, averigüe cómo puede obtener credenciales temporales para que las utilice el SDK para Java. Siga [estas instrucciones](#) para añadir credenciales temporales al archivo `~/.aws/credentials`. Si su proveedor de identidad agrega credenciales temporales automáticamente al archivo `~/.aws/credentials`, asegúrese de que el nombre del perfil sea `[default]` para que no necesite proporcionarlo al SDK o AWS CLI.

Configuración de la autenticación

En el tema [Autenticación y acceso](#) de la Guía de referencia de herramientas y AWS SDK se describen las diferentes opciones de autenticación. Le recomendamos que siga las instrucciones para [configurar el acceso al Centro de identidades de IAM para](#) que el SDK pueda adquirir las credenciales. Tras seguir las instrucciones, el [sistema estará configurado](#) para permitir que el SDK autentique las solicitudes.

Configuración para el acceso mediante inicio de sesión único para el SDK

Tras completar el paso 2 de la [sección de acceso mediante programación](#) para que el SDK pueda utilizar la autenticación del IAM Identity Center, el sistema debe contener los siguientes elementos.

- El AWS CLI, que se utiliza para iniciar una [sesión en el portal de AWS acceso](#) antes de ejecutar la aplicación.
- Un archivo `~/.aws/config` que contiene un perfil [predeterminado](#). El SDK para Java utiliza la configuración de proveedor de token de SSO del perfil para adquirir las credenciales antes de enviar las solicitudes a AWS. El `sso_role_name` valor, que es un rol de IAM conectado a un conjunto de permisos del Centro de Identidad de IAM, debería permitir el acceso a los Servicios de AWS utilizados en la aplicación.

El siguiente archivo `config` de ejemplo muestra la configuración de un perfil predeterminado con el proveedor de token de SSO. La configuración `sso_session` del perfil hace referencia a la

sección `sso-session` nombrada. La `sso-session` sección contiene la configuración para iniciar una sesión en el portal de AWS acceso.

```
[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 más detalles sobre los ajustes utilizados en la configuración del proveedor de tokens SSO, consulte [Configuración del proveedor de tokens SSO](#) en la Guía de referencia de SDK y herramientas de AWS .

Si su entorno de desarrollo no está configurado para el acceso mediante programación como se ha mostrado anteriormente, siga el [Paso 2 de la Guía de referencia de los SDK](#).

Inicie sesión con el AWS CLI

Antes de ejecutar una aplicación que acceda Servicios de AWS, necesita una sesión activa en el portal de AWS acceso para que el SDK utilice la autenticación del IAM Identity Center para resolver las credenciales. Ejecute el siguiente comando AWS CLI para iniciar sesión en el portal de AWS acceso.

```
aws sso login
```

Como tiene una configuración de perfil predeterminada, no necesita llamar al comando con una opción `--profile`. Si la configuración del proveedor de token de SSO utiliza un perfil con nombre, el comando es `aws sso login --profile named-profile`.

Para comprobar si ya tiene una sesión activa, ejecute el siguiente AWS CLI comando.

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

La respuesta a este comando debe indicar la cuenta y el conjunto de permisos del Centro de identidades de IAM configurados en el archivo compartido `config`.

Note

Si ya tiene una sesión activa en el portal de AWS acceso y la ejecuta `aws sso login`, no tendrá que proporcionar credenciales.

Sin embargo, verá un cuadro de diálogo en el que se solicita permiso para que `botocore` acceda a su información. `botocore` es la base de la AWS CLI .

Seleccione Permitir para autorizar el acceso a su información para el AWS CLI SDK for Java.

Instalar Java y una herramienta de compilación

Su entorno de desarrollo debe contar con lo siguiente:

- Java 8 o posterior AWS SDK for Java [Funciona con el kit de desarrollo Java SE de Oracle y con distribuciones del Open Java Development Kit \(OpenJDK\) Amazon Corretto, como Red Hat OpenJDK y Adoptium.](#)
- Una herramienta de compilación o IDE compatible con Maven Central, como Apache Maven, Gradle o IntelliJ.
 - [Para obtener información sobre cómo instalar y usar Maven, consulte https://maven.apache.org/.](https://maven.apache.org/)
 - Para obtener información sobre cómo instalar y usar Gradle, consulte [https://gradle.org/.](https://gradle.org/)
 - Para obtener información sobre cómo instalar y usar IntelliJ IDEA, consulte [https://www.jetbrains.com/idea/.](https://www.jetbrains.com/idea/)

Opciones de autenticación adicionales

Para obtener más opciones de autenticación para el SDK, como el uso de perfiles y variables de entorno, consulte el capítulo de [configuración](#) de la Guía de referencia de AWS SDK y herramientas.

Configurar un proyecto Apache Maven

Puede utilizar [Apache Maven](#) para configurar y compilar proyectos de AWS SDK for Java o para [compilar el propio SDK.](#)

Requisitos previos

Para usar el AWS SDK for Java con Maven, necesita lo siguiente:

- Java 8.0 o posterior. Puede descargar el software Java SE Development Kit más reciente en <http://www.oracle.com/technetwork/java/javase/downloads/>. AWS SDK for Java también funciona con [OpenJDK](#) y Amazon Corretto, una distribución del Open Java Development Kit (OpenJDK). Descargue la última versión de OpenJDK de <https://openjdk.java.net/install/index.html>. Descargue la última versión del Amazon Corretto 8 o el Amazon Corretto 11 desde [la página de Corretto](#).
- Apache Maven. Si necesita instalar Maven, vaya a <http://maven.apache.org/> para descargarlo e instalarlo.

Crear un proyecto de Maven

Para crear un proyecto Maven desde la línea de comandos, ejecute el siguiente comando desde un terminal o una ventana del indicador de comandos.

```
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

Sustituya `com.example.myapp` por el espacio de nombres del paquete completo de su aplicación. Sustituya también `myapp` por el nombre de su proyecto. Esto se convierte en el nombre del directorio del proyecto.

Para usar la última versión del arquetipo, sustituya `2.X.X` por el [último de Maven Central](#).

Este comando crea un proyecto Maven utilizando el conjunto de herramientas de plantillas de arquetipos. El arquetipo genera el andamiaje para un proyecto de controlador de funciones AWS Lambda. Este arquetipo de proyecto está preconfigurado para compilar con Java SE 8 e incluye una dependencia a la versión del SDK para Java 2.x especificada con `-DarchetypeVersion`.

Para obtener más información sobre cómo crear y configurar proyectos de Maven, consulte [Maven Getting Started Guide](#).

Configurar el compilador Java para Maven

Si ha creado su proyecto utilizando el arquetipo de proyecto AWS Lambda como se ha descrito anteriormente, la configuración del compilador Java ya está hecha por usted.

Para comprobar que esta configuración está presente, comience abriendo el archivo `pom.xml` desde la carpeta de proyecto que creó (por ejemplo, `myapp`) cuando ejecutó el comando anterior. Busque en las líneas 11 y 12 para consultar la configuración de la versión del compilador Java para este proyecto de Maven y la inclusión requerida del complemento del compilador de Maven en las líneas 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>
```

Si crea su proyecto con un arquetipo diferente o utilizando otro método, debe asegurarse de que el complemento del compilador de Maven sea parte de la compilación y que sus propiedades de origen y destino están establecidas en 1.8 en el archivo `pom.xml`.

Consulte el fragmento anterior para ver una forma de configurar estos parámetros necesarios.

Otra opción sería definir la configuración del compilador insertada con la declaración del complemento, de la siguiente manera.

```
<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 el SDK como una dependencia

Para utilizar AWS SDK for Java en el proyecto, tendrá que declararlo como una dependencia en el archivo `pom.xml` del proyecto.

Si creó su proyecto utilizando el arquetipo de proyecto como se describió anteriormente, la última versión del SDK ya está configurada como una dependencia en su proyecto.

El arquetipo genera un artefacto dependiente de la BOM (lista de materiales) para el identificador del grupo `software.amazon.awssdk`. Con una lista de materiales, no tiene que especificar la versión de Maven para las dependencias de artefactos individuales que comparten el mismo id de grupo.

Si creó su proyecto de Maven de otra manera, configure la versión más reciente del SDK para su proyecto asegurándose de que el archivo `pom.xml` contenga lo siguiente.

```
<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

Sustituya **2.X.X** en el archivo `pom.xml` por la [última versión del AWS SDK for Java 2.x](#).

Establecer dependencias para módulos del SDK

Ahora que ha configurado el SDK, puede agregar dependencias para uno o más de los módulos de AWS SDK for Java que se van a utilizar en su proyecto.

Aunque puede especificar el número de versión para cada componente, no es necesario porque ya declaró la versión del SDK en la sección `dependencyManagement` usando la lista de materiales. Para cargar una versión personalizada de un módulo determinado, especifique un número de versión para su dependencia.

Si creó su proyecto utilizando el arquetipo de proyecto como se ha descrito anteriormente, su proyecto ya está configurado con múltiples dependencias. Incluyen las dependencias de los controladores de funciones AWS Lambda y Amazon S3, de la siguiente manera.

```
<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

En el ejemplo `pom.xml` anterior, las dependencias son de diferentes `groupId`. La dependencia `s3` es de `software.amazon.awssdk`, mientras que la dependencia `aws-lambda-java-core` es de `com.amazonaws`. La configuración de administración de dependencias de la BOM afecta a los artefactos para `software.amazon.awssdk`, por lo que se necesita una versión para el artefacto `aws-lambda-java-core`.

Para el desarrollo de controladores de funciones Lambda utilizando el SDK para Java 2.x, `aws-lambda-java-core` es la dependencia correcta. Sin embargo, si la aplicación necesita administrar los recursos de Lambda, el uso de operaciones como `listFunctions`, `deleteFunction`, `invokeFunction` y `createFunction`, la aplicación requiere la siguiente dependencia.

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

Note

La dependencia `s3` excluye las dependencias transitivas `netty-nio-client` y `apache-client`. En lugar de cualquiera de esos clientes HTTP, el arquetipo incluye la dependencia `url-connection-client`, lo que ayuda a [reducir la latencia de inicio de las funciones de AWS Lambda](#).

Agregue los módulos a su proyecto para el Servicio de AWS y las características que necesita para su proyecto. Los módulos (dependencias) que administra la BOM de AWS SDK for Java se enumeran en el [repositorio central de Maven](#).

Note

Puede examinar el archivo `pom.xml` de un ejemplo de código para determinar qué dependencias necesita para su proyecto. Por ejemplo, si está interesado en las dependencias del servicio de DynamoDB, consulte [este ejemplo](#) del [Repositorio de ejemplos de código de AWS](#) en GitHub. (Busque el archivo `pom.xml` en [/java2/example_code/dynamodb.](#))

Crear todo el SDK en su proyecto

Para optimizar su aplicación, le recomendamos encarecidamente que extraiga solo los componentes que necesita en lugar de todo el SDK. Sin embargo, para crear AWS SDK for Java al completo en su proyecto, declárelo en su archivo `pom.xml` de la siguiente manera.

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

Compilación del proyecto

Después de configurar el archivo `pom.xml`, puede usar Maven para crear el proyecto.

Para crear el proyecto de Maven desde la línea de comandos, abra una ventana de terminal o símbolo del sistema, desplácese hasta el directorio del proyecto (por ejemplo, `myapp`), escriba o pegue el siguiente comando y, a continuación, presione "Enter" o "Return".

```
mvn package
```

Esto crea un único archivo `.jar` (JAR) en el directorio `target` (por ejemplo, `myapp/target`). Este JAR contiene todos los módulos del SDK especificados como dependencias en su archivo `pom.xml`.

Configurar un proyecto Gradle

Puede usar [Gradle](#) para configurar y crear proyectos AWS SDK for Java.

Los pasos iniciales del ejemplo siguiente provienen de la [Guía de introducción de Gradle](#) para la versión 8.4. Si usa una versión diferente, los resultados pueden diferir ligeramente.

Para crear una aplicación Java con Gradle (línea de comandos)

1. Cree un directorio para alojar su proyecto. En este ejemplo, el directorio se llama demo.
2. En el directorio demo, ejecute el comando `gradle init` y proporcione los valores resaltados en rojo, como se muestra en el siguiente resultado de la línea de comandos. Para explicarlo, elegimos Kotlin como lenguaje DSL para compilar scripts, pero al final de este tema también se muestra un ejemplo completo de Groovy.

```
> 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

```

- Una vez completada la tarea `init`, el directorio `demo` contiene la siguiente estructura de árbol. En la siguiente sección examinaremos más de cerca el archivo de compilación principal `build.gradle.kts` (resaltado en rojo).

```

### 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

```

El archivo `build.gradle.kts` contiene el siguiente andamiaje.

```
/*
 * 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 el archivo de compilación de Gradle como base para su proyecto AWS.
 - a. Para administrar las dependencias del SDK para su proyecto de Gradle, importe la lista de materiales de Maven (BOM) para AWS SDK for Java 2.x en la sección `dependencies` del archivo `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

En este archivo de compilación de ejemplo, sustituya 2.21.1 por la última versión del SDK para Java 2.x. Busque la última versión disponible en el [repositorio central de Maven](#).

- b. Especifique los módulos del SDK que necesita su aplicación en la sección `dependencies`. A modo de ejemplo, lo siguiente añade una dependencia de Amazon Simple Storage Service.

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

Gradle resuelve automáticamente la versión correcta de las dependencias del SDK con la información de la BOM.

Los ejemplos siguientes muestran archivos de compilación completos de Gradle en los DSL de Kotlin y Groovy. El archivo de compilación contiene las dependencias para Amazon S3, la autenticación, el registro y las pruebas. La versión de origen y destino de Java es la versión 11.

Kotlin DSL (build.gradle.kts)

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

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

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

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



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

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

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

Groovy DSL (build.gradle)

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

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

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

dependencies {
    implementation platform('software.amazon.awssdk:bom:2.21.1')
```

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

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

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

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

Para ver los pasos siguientes, consulte la guía de introducción en el sitio web de Gradle para obtener instrucciones sobre cómo [compilar y ejecutar una aplicación de Gradle](#).

Configurar un proyecto de imagen nativa de GraalVM para el AWS SDK for Java

Con las versiones 2.16.1 y posteriores, AWS SDK for Java proporciona soporte listo para usar para las aplicaciones de GraalVM Native Image. Utilice el arquetipo de archetype-app-quickstart Maven para configurar un proyecto con soporte de imágenes nativo integrado.

Requisitos previos

- Complete los pasos que se indican en [Configuración del AWS SDK for Java 2.x](#).

- Instalar [GraalVM Native Image](#).

Crear un proyecto mediante el arquetipo

Para crear un proyecto de Maven con soporte de imágenes nativo integrado, utilice el siguiente comando en una ventana de terminal o línea de comandos.

Note

Sustituya `com.example.mynativeimageapp` por el espacio de nombres completo del paquete de su aplicación. Sustituya también `mynativeimageapp` por el nombre de su proyecto. Esto se convierte en el nombre del directorio del proyecto.

```
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
```

Este comando crea un proyecto de Maven configurado con dependencias para el cliente AWS SDK for Java, Amazon S3 y el ApacheHttpClient de HTTP. También incluye una dependencia para el [complemento GraalVM Native Image Maven](#), para que pueda crear imágenes nativas con Maven.

Para incluir las dependencias de otro servicio Amazon Web Services, establezca el valor del parámetro `-Dservice` en el ID del artefacto de ese servicio. Entre los ejemplos se incluyen `dynamodb`, `comprehend` y `pinpoint`. Para obtener una lista completa de los identificadores de artefactos, consulte la lista de dependencias administradas de [software.amazon.awssdk](#) en Maven Central.

Para usar un cliente HTTP asíncrono, defina el parámetro `-DhttpClient` como `netty-nio-client`. Para usar `URLConnectionHttpClient` como cliente HTTP asíncrono en lugar de `apache-client`, defina el parámetro `-DhttpClient` en `url-connection-client`.

Crear una imagen nativa

Tras crear el proyecto, ejecute el siguiente comando desde el directorio del proyecto, por ejemplo `mynativeimageapp`:

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

Esto crea una aplicación de imagen nativa en el directorio `target`, por ejemplo, `target/mynativeimageapp`.

Usa el AWS SDK for Java 2.x

Tras completar los pasos de [Configuración del SDK](#), estará listo para realizar solicitudes a AWS servicios como Amazon S3, DynamoDB, IAM, Amazon EC2 y más.

Trabajar con los clientes de servicios

Crear un cliente de servicio

Para realizar una solicitud a un Servicio de AWS, primero debe crear una instancia de un cliente de servicio para ese servicio mediante el método estático de fábrica, `builder()`. El método `builder()` devuelve un objeto `builder` que permite personalizar el cliente de servicio. Los métodos `setter` `Fluent` devuelven el objeto `builder` para que pueda encadenar fácilmente las llamadas a los métodos y para simplificar la lectura del código. Después de configurar las propiedades que desee, puede llamar al método `build()` para crear el cliente.

A modo de ejemplo, el siguiente fragmento de código crea una instancia de un objeto `Ec2Client` como cliente de servicio para Amazon EC2.

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

Note

Los clientes de servicios del SDK son seguros para subprocessos. Para obtener el máximo desempeño, trátelos como objetos de larga duración. Cada cliente tiene su propio recurso de grupo de conexiones que se libera cuando el cliente recopila los elementos no utilizados. Un objeto cliente de servicio es inmutable, por lo que deberá crear un nuevo cliente para cada servicio al que haga peticiones, o si desea utilizar una configuración diferente para realizar peticiones al mismo servicio.

No es necesario especificarlo `Region` en el generador de clientes de AWS servicios para todos los servicios; sin embargo, se recomienda establecer la región de las llamadas a la API que realice en sus aplicaciones. Para obtener más información, consulte la [Selección de regiones de AWS](#).

Configuración predeterminada de cliente

Los creadores de clientes tienen otro método de fábrica denominado `create()`. Este método crea un servicio cliente con la configuración predeterminada. Utiliza la cadena de proveedores predeterminada para cargar las credenciales y la Región de AWS. Si las credenciales o la región no se pueden determinar a partir del entorno en el que se ejecuta la aplicación, la llamada a `create` produce un error. Consulte [Uso de credenciales](#) y [Selección de regiones](#) para obtener más información acerca de cómo el SDK determinan las credenciales y la región.

A modo de ejemplo, el siguiente fragmento de código crea una instancia de un objeto `DynamoDbClient` como cliente de servicio para Amazon DynamoDB:

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

Configurar clientes de servicio

Para personalizar la configuración de un cliente de servicio, utilice los parámetros del método de fábrica `builder()`. Para mayor comodidad y para crear un código más legible, encadene los métodos para establecer varias opciones de configuración.

El siguiente ejemplo muestra un `S3Client` configurado con varios ajustes personalizados.

```
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();
```

Hacer solicitudes

Utilice el cliente de servicio para realizar solicitudes al correspondiente Servicio de AWS.

Por ejemplo, este fragmento de código muestra cómo crear un objeto de `RunInstancesRequest` para crear una nueva instancia de 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);
```

En lugar de crear una solicitud y pasarla a la instancia, el SDK proporciona creadores que puede usar para crear una solicitud. Con un creador, puede usar expresiones lambda de Java para crear la solicitud “en línea”.

En el siguiente ejemplo, se reescribe el ejemplo anterior mediante la versión del `runInstances` [método que utiliza un creador](#) para crear la solicitud.

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

Tratamiento de respuestas

El SDK devuelve un objeto de respuesta para la mayoría de las operaciones de servicio. El código puede procesar la información del objeto de respuesta según sus necesidades.

Por ejemplo, el siguiente fragmento de código imprime el identificador de la primera instancia devuelto con el [RunInstancesResponse](#) objeto en la solicitud anterior.

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

Sin embargo, no todas las operaciones devuelven un objeto de respuesta con datos específicos del servicio. En estas situaciones, puede consultar el estado de la respuesta HTTP para saber si la operación se realizó correctamente.

Por ejemplo, el código del siguiente fragmento comprueba la respuesta HTTP para comprobar si la [DeleteContactList](#) operación de Amazon Simple Email Service se ha realizado correctamente.

```
SesV2Client sesv2Client = SesV2Client.create();

DeleteContactListRequest request = DeleteContactListRequest.builder()
    .contactListName("ExampleContactListName")
    .build();

DeleteContactListResponse response = sesv2Client.deleteContactList(request);
if (response.sdkHttpResponse().isSuccessful()) {
    System.out.println("Contact list deleted successfully");
} else {
    System.out.println("Failed to delete contact list. Status code: " +
        response.sdkHttpResponse().statusCode());
}
```

Cerrar el cliente de servicio

Como práctica recomendada, debe utilizar un cliente de servicio para varias llamadas al servicio de la API durante la vida útil de una aplicación. Sin embargo, si necesita un cliente de servicio para un solo uso o ya no lo necesita, ciérrelo.

Para liberar recursos, llame al método `close()` cuando el cliente de servicio deje de ser necesario.

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

Si necesita un cliente de servicio para un solo uso, puede crear una instancia del cliente de servicio como un recurso en una instrucción `try-with-resources`. Los clientes de servicio implementan la interfaz de [Autoclosable](#), por lo que el JDK llama automáticamente al método `close()` al final de la instrucción.

En el ejemplo siguiente, se muestra cómo utilizar un cliente de servicio para una llamada única. El `StsClient` que llama al AWS Security Token Service se cierra después de devolver el ID de la cuenta.

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

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

Tratamiento de excepciones

El SDK utiliza excepciones en tiempo de ejecución (o no comprobadas), lo que te proporciona un control pormenorizado sobre el tratamiento de errores y garantiza que el tratamiento de excepciones se adapte a su aplicación.

Una [SdkServiceException](#), o una de sus subclases, es la forma de excepción más común que generará el SDK. Estas excepciones representan las respuestas del servicio de AWS. También puede tratar una [SdkClientException](#), que se produce cuando hay un problema en el cliente (es decir, en el entorno de desarrollo o de aplicaciones), como un fallo de conexión de red.

Este fragmento de código muestra una forma de gestionar las excepciones de servicio al cargar un archivo en Amazon S3. El código de ejemplo captura las excepciones del cliente y del servidor, registra los detalles y sale de la aplicación.

```
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);
}
```

Consulte [Tratamiento de excepciones](#) para obtener más información.

Utilizar esperadores

El procesamiento de algunas solicitudes lleva tiempo, como la creación de una nueva tabla DynamoDB o la creación de un nuevo Amazon S3 depósito. Para asegurarse de que el recurso esté listo antes de que el código siga ejecutándose, use un esperador.

Por ejemplo, este fragmento de código crea una tabla nueva («MyTable») en DynamoDB, espera a que la tabla tenga un ACTIVE estado y, a continuación, imprime la respuesta:

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

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

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

Consulte [Uso de los esperadores](#) para obtener más información.

Cientes de HTTP

Puede cambiar la configuración predeterminada de los clientes HTTP en las aplicaciones que compile con el AWS SDK for Java. Para obtener información sobre cómo configurar los clientes y las opciones HTTP, consulte [Configuración HTTP](#).

Tiempos de espera

Puede configurar los tiempos de espera para cada uno de sus clientes de servicio mediante los valores `apiCallTimeout` y los valores de.

[apiCallAttemptTimeoutClientOverrideConfiguration.Builder](#) La configuración `apiCallTimeout` es la cantidad de tiempo que se tarda en permitir que el cliente complete la ejecución de una llamada a la API. La `apiCallAttemptTimeout` configuración es el tiempo que se debe esperar a que se complete cada solicitud HTTP (reintento) antes de darse por vencida.

El siguiente ejemplo establece ambos tiempos de espera para un cliente S3.

```
S3Client s3Client = S3Client.builder()
    .overrideConfiguration(b -> b
        .apiCallTimeout(Duration.ofSeconds(105L))
        .apiCallAttemptTimeout(Duration.ofSeconds(25L)))
    .build();
```

También puede establecer los tiempos de espera en el nivel de solicitud configurando un [AwsRequestOverrideConfiguration](#) y proporcionándolo al objeto de solicitud con el `overrideConfiguration` método.

En el siguiente ejemplo, se utiliza la misma configuración de tiempo de espera, pero en el nivel de solicitud para una operación de `S3PutObject`.

```
S3Client basicS3Client = S3Client.create(); // Client with no timeout settings.

AwsRequestOverrideConfiguration overrideConfiguration =
    AwsRequestOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofSeconds(105L))
        .apiCallAttemptTimeout(Duration.ofSeconds(25L))
        .build();

basicS3Client.putObject(b -> b
    .bucket("DOC-EXAMPLE-BUCKET")
    .key("DOC-EXAMPLE_KEY")
    .overrideConfiguration(overrideConfiguration),
    RequestBody.fromString("test"));
```

Interceptores de ejecución

Puede escribir código que intercepte la ejecución de las solicitudes y respuestas de la API en distintas partes del ciclo de vida de la solicitud/respuesta. Esto permite publicar métricas, modificar una solicitud durante el proceso, depurar el procesamiento de las solicitudes, ver las excepciones y mucho más. Para obtener más información, consulte [la `ExecutionInterceptor` interfaz](#) en la referencia de la AWS SDK for Java API.

Información adicional

- Para ver ejemplos completos de los fragmentos de código anteriores, consulte [Trabajar con Amazon DynamoDB](#), [Trabajar con Amazon EC2](#) y [Trabajar con Amazon S3](#)

Proporcionar credenciales temporales al SDK

Antes de solicitar su Amazon Web Services uso AWS SDK for Java 2.x, el SDK firma criptográficamente las credenciales temporales emitidas por AWS. Para acceder a las credenciales temporales, el SDK recupera los valores de configuración comprobando varias ubicaciones.

En este tema se describen varias formas de permitir que el SDK acceda a las credenciales temporales.

Temas

- [Configurar el acceso a las credenciales temporales](#)
- [Cadena predeterminada de proveedores de credenciales](#)
- [Usar un determinado proveedor de credenciales o una cadena de proveedores](#)
- [Perfiles de usuario](#)
- [Cargar las credenciales temporales de un proceso externo](#)
- [Proporcione las credenciales temporales mediante código](#)
- [Lea las credenciales del rol de IAM en Amazon EC2](#)

Configurar el acceso a las credenciales temporales

Para aumentar la seguridad, se AWS recomienda configurar el SDK para Java para que [utilice credenciales temporales en lugar de credenciales](#) de larga duración. Las credenciales temporales

consisten en claves de acceso (id de clave de acceso y clave de acceso secreta) y un token de sesión. Le recomendamos que [configure el SDK](#) para que obtenga automáticamente las credenciales temporales, ya que el proceso de actualización del token es automático. Sin embargo, puede [proporcionar credenciales temporales directamente al SDK](#).

Configuración del Identity Center de IAM

Al configurar el SDK para utilizar el acceso de inicio de sesión único al IAM Identity Center, tal como se describe en [???](#) de esta guía, el SDK utiliza automáticamente las credenciales temporales.

El SDK utiliza el token de acceso al IAM Identity Center para acceder al rol de IAM definido con la configuración `sso_role_name` del archivo `config`. El SDK asume este rol de IAM y recupera las credenciales temporales para usarlas en las solicitudes de Servicio de AWS .

Para obtener más información sobre cómo el SDK obtiene las credenciales temporales de la configuración, consulte la sección [Descripción de la autenticación del IAM Identity Center](#) de la Guía de referencia de AWS los SDK y las herramientas.

Recuperar desde el portal de acceso de AWS

Como alternativa a la configuración de inicio de sesión único del IAM Identity Center, puede copiar y utilizar las credenciales temporales disponibles en el portal de acceso. AWS Puede utilizar las credenciales temporales en un perfil o como valores para las propiedades del sistema y las variables de entorno.

Configurar un archivo local de credenciales para las credenciales temporales

1. [Crear un archivo de credenciales compartido](#).
2. En el archivo de credenciales, pegue el siguiente texto de marcador de posición hasta que pegue las credenciales temporales que funcionen.

```
[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. Guarde el archivo. El archivo `~/.aws/credentials` debería existir ahora en su sistema de desarrollo local. Este archivo contiene el [perfil \[predeterminado\]](#) que el SDK para Java utiliza si no se especifica un perfil con nombre específico.
4. [Inicie sesión en el portal de acceso AWS](#)


```
DynamoDbClient ddb =
    DynamoDbClient.builder()
        .region(region)
        .build();
```

Orden de recuperación de la configuración de credenciales

La cadena predeterminada de proveedores de credenciales del SDK para Java 2.x busca la configuración en su entorno mediante una secuencia predefinida.

1. Propiedades del sistema Java

- El SDK usa la [SystemPropertyCredentialsProvider](#) clase para cargar credenciales temporales desde las propiedades del sistema `aws.accessKeyId`, `aws.secretAccessKey`, y `aws.sessionToken` Java.

Note

Para obtener información sobre cómo configurar las propiedades del sistema Java, consulte el tutorial Propiedades del sistema en el sitio web oficial de [tutoriales de Java](#).

2. Variables de entorno

- El SDK usa la [EnvironmentVariableCredentialsProvider](#) clase para cargar credenciales temporales de las variables de `AWS_SESSION_TOKEN` entorno `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, y.

3. Token de identidad web de AWS Security Token Service

- El SDK usa la [WebIdentityTokenFileCredentialsProvider](#) clase para cargar credenciales temporales desde las propiedades del sistema Java o las variables de entorno.

4. Los archivos compartidos `credentials` y `config`

- El SDK las utiliza [ProfileCredentialsProvider](#) para cargar la configuración de inicio de sesión único del IAM Identity Center o las credenciales temporales del [default] perfil en los archivos `AND` compartidos `credentials.config`

La guía de referencia de herramientas y AWS SDK contiene [información detallada](#) sobre cómo funciona el SDK para Java con el token de inicio de sesión único del IAM Identity Center para obtener las credenciales temporales que el SDK utiliza para realizar llamadas. Servicios de AWS

Note

Varios `credentials` AWS SDK y config herramientas comparten los archivos y. Para obtener más información, consulte [Archivos `.aws/credentials` y `.aws/config`](#) en la Guía de referencia de SDK y herramientas de AWS .

5. Amazon ECS credenciales de contenedor

- El SDK usa la [ContainerCredentialsProvider](#) clase para cargar credenciales temporales desde la variable de entorno `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` del sistema.

6. Amazon EC2 credenciales proporcionadas por el rol de IAM de la instancia

- El SDK usa la [InstanceProfileCredentialsProvider](#) clase para cargar credenciales temporales desde el Amazon EC2 servicio de metadatos.

Usar un determinado proveedor de credenciales o una cadena de proveedores

Como alternativa a la cadena de proveedores de credenciales predeterminada, puede especificar qué proveedor de credenciales debe usar el SDK. Al proporcionar un proveedor de credenciales específico, el SDK omite el proceso de comprobar varias ubicaciones, lo que reduce ligeramente el tiempo necesario para crear un cliente de servicio.

Por ejemplo, si estableces tu configuración predeterminada mediante variables de entorno, proporciona un [EnvironmentVariableCredentialsProvider](#) objeto al `credentialsProvider` método en el generador de clientes del servicio, como se muestra en el siguiente fragmento de código.

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

Para obtener una lista completa de los proveedores de credenciales y las cadenas de proveedores, consulte Todas las clases de implementación conocidas en [AwsCredentialsProvider](#)

Note

Puede usar su propio proveedor de credenciales o cadenas de proveedores implementando la interfaz de `AwsCredentialsProvider`.

Perfiles de usuario

Utilizando el archivo compartido de `credentials` y `config`, puede configurar varios perfiles. Esto posibilita que su aplicación utilice varios conjuntos de configuración de credenciales. El perfil `[default]` se mencionó anteriormente. El SDK usa la [ProfileCredentialsProvider](#) clase para cargar la configuración de los perfiles definidos en el `credentials` archivo compartido.

El siguiente fragmento de código muestra cómo crear un cliente de servicio que utilice la configuración definida como parte del perfil denominado `my_profile`.

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

Establecer un perfil predeterminado diferente

Para establecer como predeterminado un perfil distinto del perfil `[default]` para su aplicación, defina la variable de entorno `AWS_PROFILE` con el nombre de su perfil personalizado.

Para establecer esta variable en Linux, MacOS, o Unix, utilice `export`:

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

Para establecer estas variables en Windows, utilice `set`:

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

Como alternativa, defina la propiedad del sistema Java `aws.profile` con el nombre del perfil.

Volver a cargar credenciales del perfil

Puede configurar cualquier proveedor de credenciales que tenga un método `profileFile()` en su generador para volver a cargar las credenciales del perfil. Estas clases de perfiles de credenciales son: `ProfileCredentialsProvider`, `DefaultCredentialsProvider`, `InstanceProfileCredentialsProvider` y `ProfileTokenProvider`.

Note

La recarga de credenciales de perfil solo funciona con los siguientes ajustes en el archivo de perfil: `aws_access_key_id`, `aws_secret_access_key` y `aws_session_token`. Los ajustes de `region`, `sso_session`, `sso_account_id` y `source_profile` no se tienen en cuenta.

Para configurar un proveedor de credenciales compatible para volver a cargar la configuración del perfil, proporcione una instancia del [ProfileFileSupplier](#) al método de creación del `profileFile()`. El siguiente ejemplo de código demuestra un `ProfileCredentialsProvider` que recarga la configuración de credenciales desde el 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().
*/
```

Cuando se llama a `ProfileCredentialsProvider.resolveCredentials()`, el SDK para Java vuelve a cargar la configuración. `ProfileFileSupplier.defaultSupplier()` es una de las [diversas implementaciones prácticas](#) de `ProfileFileSupplier` ofrecidas por el SDK. Si su caso lo requiere, puede proporcionar su propia implementación.

En el ejemplo siguiente, se muestra el uso del método de conveniencia `ProfileFileSupplier.reloadWhenModified()`. `reloadWhenModified()` toma un parámetro `Path`, lo que le da flexibilidad a la hora de designar el archivo de origen de la configuración en lugar de la ubicación estándar `~/.aws/credentials` (o `config`).

La configuración se volverá a cargar cuando `resolveCredentials()` se invoque solo si el SDK determina que el contenido del archivo se ha 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.
*/
```

El método `ProfileFileSupplier.aggregate()` combina el contenido de varios archivos de configuración. Usted decide si un archivo se vuelve a cargar por cada llamada a `resolveCredentials()` o si la configuración de un archivo se fija en el momento en que se leyó por primera vez.

El siguiente ejemplo muestra un `DefaultCredentialsProvider` que combina la configuración de dos archivos que contienen la configuración del perfil. El SDK vuelve a cargar la configuración del archivo al que apunta la variable `credentialsFilePath` cada vez que se llama a `resolveCredentials()` y cambia la configuración. La configuración del objeto `profileFile` sigue siendo la misma.

```
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.
*/
```

Cargar las credenciales temporales de un proceso externo

Warning

A continuación se describe un método para obtener credenciales de un proceso externo. Esto resulta potencialmente peligroso, así que proceda con precaución. Si es posible, se debe dar preferencia a otros proveedores de credenciales. Si usa esta opción, debe asegurarse de que el archivo `config` esté lo más bloqueado posible siguiendo las mejores prácticas de seguridad para su sistema operativo.

Asegúrese de que su herramienta de credenciales personalizadas no escriba ninguna información secreta en ella `StdErr`. Es un SDK y AWS CLI puede capturar y registrar dicha información, lo que podría exponerla a usuarios no autorizados.

Con el SDK para Java 2.x, puede adquirir credenciales temporales de un proceso externo para casos de uso personalizados. Hay dos formas de configurar esta funcionalidad.

Usar la configuración de **credential_process**

Si tiene un método que proporciona credenciales temporales, puede integrarlo añadiendo la configuración de `credential_process` como parte de una definición de perfil en el archivo `config`. El valor que especifique debe utilizar la ruta completa al archivo de comandos. Si la ruta del archivo contiene espacios, debe escribirla entre comillas.

El SDK llama al comando exactamente como se especifica y, a continuación, lee los datos JSON desde `stdout`.

Los ejemplos siguientes muestran el uso de esta configuración para las rutas de archivos sin espacios y las rutas de archivos con espacios.

Linux/macOS

Sin espacios en la ruta del archivo

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

Con espacios en la ruta del archivo

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

Windows

Sin espacios en la ruta del archivo

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

Con espacios en la ruta del archivo

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

El siguiente fragmento de código muestra cómo crear un cliente de servicio que utilice la configuración definida como parte del perfil denominado `process-credential-profile`.

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

Para obtener información detallada sobre el uso de un proceso externo como fuente de credenciales temporales, consulta la [sección sobre credenciales de procesos](#) de la Guía de referencia de AWS SDK y herramientas.

Utilizar la **ProcessCredentialsProvider**

Como alternativa al uso de la configuración del archivo config, puede usar los [ProcessCredentialsProvider](#) del SDK para cargar credenciales temporales mediante Java.

Los siguientes ejemplos muestran varias versiones de cómo especificar un proceso externo utilizando el `ProcessCredentialsProvider` y configurando un cliente de servicio que utilice las credenciales temporales.

Linux/macOS

Sin espacios en la ruta del archivo

```
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();
```

Con espacios en la ruta del archivo

```
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

Sin espacios en la ruta del archivo

```
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();
```

Con espacios en la ruta del archivo

```
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();
```

Proporcione las credenciales temporales mediante código

Si la cadena de credenciales predeterminada o un proveedor o cadena de proveedores específicos o personalizados no funcionan para su código, puede proporcionar mediante código las credenciales temporales que desee. Pueden ser [credenciales de rol de IAM](#), tal como [se describió anteriormente](#), o credenciales temporales recuperadas de AWS Security Token Service (AWS STS). Si ha obtenido credenciales temporales utilizando AWS STS, entréguelas a un Servicio de AWS cliente, tal y como se muestra en el siguiente ejemplo de código.

1. Asuma un rol llamando a `StsClient.assumeRole()`.
2. Cree un [StaticCredentialsProvider](#) objeto y suministre el `AwsSessionCredentials` objeto.

3. Configure el compilador del cliente con `StaticCredentialsProvider` y compile el cliente.

En el siguiente ejemplo, se crea un cliente de servicio Amazon S3 con las credenciales temporales devueltas AWS STS por un rol asumido por IAM.

```
// The AWS IAM Identity Center identity (user) who executes this method does not
// have permission to list buckets.
// The identity is configured in the [default] profile.
public static void assumeRole(String roleArn, String roleSessionName) {
    // The IAM role represented by the 'roleArn' parameter can be assumed by
    // identities in two different accounts
    // and the role permits the user to only list buckets.

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

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

        // List all buckets in the account associated with the assumed role
        // by using the temporary credentials retrieved by invoking
        stsClient.assumeRole().
        StaticCredentialsProvider staticCredentialsProvider =
        StaticCredentialsProvider.create(
            AwsSessionCredentials.create(key, secKey, secToken));
        try (S3Client s3 = S3Client.builder()
            .credentialsProvider(staticCredentialsProvider)
            .build()) {
            List<Bucket> buckets = s3.listBuckets().buckets();
        }
    }
}
```



```
        for (Bucket bucket : buckets) {
            System.out.println("bucket name: " + bucket.name());
        }
    }
} catch (StsException | S3Exception e) {
    logger.error(e.getMessage());
    System.exit(1);
}
}
```

Conjunto de permisos

El siguiente conjunto de permisos, definido en AWS IAM Identity Center permite a la identidad (usuario) hacer las dos operaciones siguientes

1. La `GetObject` operación del Amazon Simple Storage Service.
2. La operación `AssumeRole` del AWS Security Token Service.

Sin asumir el rol, fallaría el método `s3.listBuckets()` que se muestra en el ejemplo.

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

Rol asumido

Política de permisos del rol asumidos

La política de permisos siguiente está asociada al rol que se asume en el ejemplo anterior. Esta política de permisos permite enumerar todos los buckets de la misma cuenta que el rol.

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

Política de confianza del rol asumido

La política de permisos siguiente está asociada al rol que se asume en el ejemplo anterior. La política permite que las identidades (usuarios) de dos cuentas asuman el rol.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:root",
          "arn:aws:iam::555555555555:root"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

Lea las credenciales del rol de IAM en Amazon EC2

Puede utilizar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una instancia EC2 y que realizan solicitudes a la API AWS CLI . AWS Es preferible

hacerlo de este modo a almacenar claves de acceso en la instancia EC2. Para asignar un AWS rol a una instancia EC2 y ponerlo a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia EC2 obtener credenciales temporales. Para más información, consulte [Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2](#) en la Guía del usuario de IAM.

En este tema se proporciona información sobre cómo configurar la aplicación Java para que se ejecute en una instancia EC2 y permitir que el SDK para Java adquiera credenciales de IAM rol.

Adquiera las credenciales de rol de IAM del entorno

Si la aplicación crea un cliente de AWS servicio mediante el `create` método (o `builder().build()` los métodos), el SDK para Java utiliza la cadena de proveedores de credenciales predeterminada. La cadena de proveedores de credenciales predeterminada busca en el entorno de ejecución elementos de configuración que el SDK pueda intercambiar por credenciales temporales. En la sección [the section called “Cadena predeterminada de proveedores de credenciales”](#) se describe el proceso de búsqueda completo.

El último paso de la cadena de proveedores predeterminada solo está disponible cuando la aplicación se ejecuta en una Amazon EC2 instancia. En este paso, el SDK utiliza un `InstanceProfileCredentialsProvider` para leer el rol de IAM definido en el perfil de la instancia EC2. A continuación, el SDK adquiere las credenciales temporales para ese rol de IAM.

Aunque estas credenciales son temporales y acaban caducando, un `InstanceProfileCredentialsProvider` las actualiza periódicamente para que sigan permitiendo el acceso a AWS.

Adquiera las credenciales de los roles de IAM mediante programación

Como alternativa a la cadena de proveedores de credenciales predeterminada, que eventualmente utiliza un `InstanceProfileCredentialsProvider` EC2, puede configurar un cliente de servicio de forma explícita con un `InstanceProfileCredentialsProvider`. Este método se muestra en el fragmento de código siguiente.

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

Adquiera credenciales de rol de IAM de forma segura

De forma predeterminada, las instancias EC2 ejecutan el [IMDS](#) (Instance Metadata Service), que permite a los SDK acceder `InstanceProfileCredentialsProvider` a información como la función de IAM que se ha configurado. Las instancias EC2 ejecutan dos versiones del IMDS de forma predeterminada:

- Servicio de metadatos de instancia, versión 1 (IMDSv1): un método de solicitud y respuesta
- Servicio de metadatos de instancia, versión 2 (IMDSv2): un método orientado a la sesión

[IMDSv2 es un enfoque más seguro](#) que IMDSv1.

De forma predeterminada, el SDK de Java primero intenta con IMDSv2 para obtener la función de IAM, pero si no lo consigue, prueba con IMDSv1. Sin embargo, dado que IMDSv1 es menos seguro, se AWS recomienda usar solo IMDSv2 e impedir que el SDK pruebe IMDSv1.

Para usar un enfoque más seguro, deshabilite el uso de IMDSv1 por parte del SDK proporcionando una de las siguientes configuraciones con un valor de `true`

- Variable de entorno: `AWS_EC2_METADATA_V1_DISABLED`
- Propiedad del sistema JVM: `aws.disableEc2MetadataV1`
- Configuración del archivo de configuración compartido: `ec2_metadata_v1_disabled`

Si uno de estos ajustes está establecido en `true`, el SDK no carga las credenciales del rol de IMDS mediante IMDSv1 si se produce un error en la llamada inicial de IMDSv2.

Uso Regiones de AWS

Regiones de AWS permiten a los clientes del servicio acceder a lugares Servicios de AWS que se encuentran físicamente en un área geográfica específica.

Configure explícitamente una Región de AWS

Para configurar de forma explícita una región, le recomendamos que utilice las constantes definidas en la clase [Region](#). Esta es una enumeración de todas las regiones disponibles públicamente.

Para crear un cliente con una región enumerada de la clase, utilice el método `region` del creador de clientes.

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

Si la región que intenta utilizar no es una de las enumeraciones en la clase `Region`, puede crear una nueva región con el método `of` estático. Este método le permite obtener acceso a nuevas regiones sin actualizar el SDK.

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

Note

Después de crear un cliente con el creador, es inmutable y Región de AWS no se puede cambiar. Si necesita trabajar con varios Regiones de AWS para el mismo servicio, debe crear varios clientes, uno por región.

Dejar que el SDK determine automáticamente la región desde el entorno

Cuando el código se ejecute en Amazon EC2 o AWS Lambda, tal vez, desee configurar los clientes para que usen los mismos en los Región de AWS que se ejecuta el código. Esto desvincula el código del entorno en el que se ejecuta y facilita la implementación de la aplicación en varios niveles Regiones de AWS para reducir la latencia o la redundancia.

Para utilizar la cadena predeterminada de proveedores de credenciales o regiones para determinar la región a partir del entorno, use el método `create` del creador del cliente.


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

Si no configuras una de forma explícita Región de AWS mediante el `region` método, el SDK consulta la cadena de proveedores de regiones predeterminada para determinar la región que se va a utilizar.

Comprensión de la cadena de proveedores de la región predeterminada

El SDK sigue los pasos indicados a continuación para buscar una Región de AWS :

1. Cualquier región explícita establecida mediante `region` en el propio creador prevalece sobre todas las demás.
2. Se comprueba la variable de entorno `AWS_REGION`. Si se ha establecido, se usa esa región para configurar el cliente.

 Note

El Lambda contenedor establece esta variable de entorno.

3. El SDK comprueba el archivo de configuración AWS compartido y el archivo de credenciales compartidas (normalmente ubicados en `~/.aws/config` y `~/.aws/credentials`). Si la `region` propiedad está presente, el SDK la usa.
 - Si el SDK encuentra la `region` propiedad en ambos archivos del mismo perfil (incluido el `default` perfil), usará el valor del archivo de credenciales compartido.
 - La variable de entorno `AWS_CONFIG_FILE` se puede utilizar para personalizar la ubicación del archivo de configuración compartida.
 - La variable de entorno `AWS_PROFILE` o la propiedad del sistema `aws.profile` se pueden utilizar para especificar el perfil que carga el SDK.
4. El SDK intenta usar el servicio de metadatos de la Amazon EC2 instancia (IMDS) para determinar la región de la Amazon EC2 instancia que se está ejecutando actualmente.
 - Para mayor seguridad, debes deshabilitar el SDK para que no intente usar la versión 1 del IMDS. Para deshabilitar la versión 1, utilice la misma configuración que se describe en la [the section called “De forma segura”](#) sección.
5. Si el SDK todavía no ha encontrado una región en ese momento, la creación del cliente produce una excepción.

Al desarrollar AWS aplicaciones, un enfoque habitual consiste en utilizar el archivo de configuración compartido (descrito en [Orden de recuperación de credenciales](#)) para configurar la región para el desarrollo local y confiar en la cadena de proveedores de la región predeterminada para determinar la región cuando la aplicación se ejecuta en AWS la infraestructura. Esto simplifica enormemente la creación del cliente y dota de portabilidad a su aplicación.

Comprobación de la disponibilidad del servicio en una región

Para comprobar si una determinada región Servicio de AWS está disponible en una región, utilice el `region` método `serviceMetadata` y en el cliente del servicio.

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

Consulte la documentación de la clase [Region](#) para ver Regiones de AWS lo que puede especificar y utilice el prefijo de punto final del servicio para realizar la consulta.

Seleccionar un punto de conexión específico

En determinadas situaciones (por ejemplo, para probar una vista previa de las funciones de un servicio antes de que pasen a estar disponibles para el público general), es posible que tenga que especificar un punto de conexión específico en una región. En estas situaciones, los clientes del servicio se pueden configurar llamando al método. `endpointOverride`

Por ejemplo, para configurar un Amazon EC2 cliente para que utilice la región de Europa (Irlanda) con un punto final específico, utilice el siguiente código.

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

Consulte [Regiones y puntos de enlace](#) para ver la lista actual de regiones y sus puntos de enlace correspondientes para todos los AWS servicios.

Reduzca el tiempo de inicio del SDK para AWS Lambda

Uno de los objetivos del AWS SDK for Java 2.x es reducir la latencia de inicio de las AWS Lambda funciones. El SDK contiene cambios que reducen el tiempo de startup, descritos al final de este tema.

En primer lugar, este tema se centra en los cambios que puede realizar para reducir los tiempos de arranque en frío. Esto incluye realizar cambios en la estructura de su código y en la configuración de los clientes del servicio.

Utilice un cliente HTTP AWS basado en CRT

Para trabajar con él AWS Lambda, recomendamos el [AwsCrtHttpClient](#) para escenarios sincrónicos y [AwsCrtAsyncHttpClient](#) asíncronos.

En el [the section called “Configurar clientes AWS HTTP basados en CRT”](#) tema de esta guía se describen las ventajas de utilizar los clientes HTTP, cómo añadir la dependencia y cómo configurar su uso por parte de los clientes de servicio.

Eliminar las dependencias del cliente HTTP no utilizadas

Además del uso explícito de un cliente AWS basado en CRT, puedes eliminar otros clientes HTTP que el SDK incluye de forma predeterminada. El tiempo de startup de Lambda se reduce cuando se necesita cargar menos librerías, por lo que debería eliminar cualquier artefacto no utilizado que la JVM necesite cargar.

El siguiente fragmento de código de un archivo `pom.xml` Maven muestra la exclusión del cliente HTTP basado en Apache y el cliente HTTP basado en Netty. (Estos clientes no son necesarios cuando se utiliza un cliente basado en AWS CRT). En este ejemplo, se excluyen los artefactos del cliente HTTP de la dependencia del cliente S3 y se agrega el `aws-crt-client` artefacto para permitir el acceso a los AWS clientes HTTP basados en CRT.

```
<project>
  <properties>
    <aws.java.sdk.version>2.25.51</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>aws-crt-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>
```

Note

Añada el elemento `<exclusions>` a todas las dependencias del cliente de servicio del archivo `pom.xml`.

Configurar los clientes de servicio para abreviar las búsquedas

Especificar una región

Cuando cree un cliente de servicio, llame al método `region` en el generador del cliente de servicio. Esto reduce el [proceso de búsqueda de regiones](#) predeterminado del SDK, que busca la información en varios lugares. Región de AWS

Para mantener el código Lambda independiente de la región, utilice el siguiente código dentro del método `region`. Este código accede a la variable de entorno `AWS_REGION` establecida por el contenedor Lambda.

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

Utilizar la **EnvironmentVariableCredentialProvider**

Al igual que el comportamiento de búsqueda predeterminado para la información de la Región, el SDK busca las credenciales en varios lugares. Si especifica el [EnvironmentVariableCredentialProvider](#) al generar un cliente de servicio, ahorrará tiempo en el proceso de búsqueda del SDK.

Note

El uso de este proveedor de credenciales permite que el código se utilice en Lambda funciones, pero es posible que no funcione en Amazon EC2 otros sistemas.

El siguiente fragmento de código muestra un cliente de servicio S3 configurado adecuadamente para su uso en un entorno Lambda.

```
S3Client s3Client = S3Client.builder()

    .region(Region.of(System.getenv(SdkSystemSetting.AWS_REGION.environmentVariable())))
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .httpClient(AwsCrtHttpClient.builder().build())
    .build();
```

Inicializar el cliente de SDK fuera del controlador de la función de Lambda

Recomendamos inicializar un cliente SDK fuera del método del controlador Lambda. De esta forma, si se reutiliza el contexto de ejecución, se puede omitir la inicialización del cliente de servicio. Al reutilizar la instancia del cliente y sus conexiones, las invocaciones posteriores al método controlador se producen más rápidamente.

En el siguiente ejemplo, la instancia `S3Client` se inicializa en el generador mediante un método de fábrica estático. Si se reutiliza el contenedor que administra el entorno Lambda, se reutiliza la instancia `S3Client` inicializada.

```
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.
    }
}
```

Minimizar la inyección de dependencias

Los marcos de inyección de dependencias (DI) pueden tardar más tiempo en completar el proceso de configuración. También es posible que requieran dependencias adicionales, que tardan un tiempo en cargarse.

Si se necesita un marco de DI, recomendamos usar marcos DI livianos como [Dagger](#).

Usa un arquetipo de segmentación de Maven AWS Lambda

El equipo del SDK de AWS Java ha desarrollado una plantilla [Maven Archetype](#) para arrancar un proyecto Lambda con un tiempo de inicio mínimo. Puede crear un proyecto de Maven a partir del arquetipo y saber que las dependencias están configuradas adecuadamente para el entorno de Lambda.

Para obtener más información sobre el arquetipo y trabajar con un ejemplo de implementación, consulte esta [entrada del blog](#).

Considere Lambda SnapStart para Java

Si sus requisitos de tiempo de ejecución son compatibles, AWS ofrece [Lambda SnapStart para Java](#). Lambda SnapStart es una solución basada en infraestructura que mejora el rendimiento inicial de las funciones de Java. Al publicar una nueva versión de una función, Lambda la SnapStart inicializa y toma una instantánea cifrada e inmutable del estado de la memoria y del disco. SnapStart a continuación, guarda en caché la instantánea para volver a utilizarla.

Cambios en la versión 2.x que afectan al tiempo de startup

Además de los cambios que haga en su código, la versión 2.x del SDK para Java incluye tres cambios principales que reducen el tiempo de startup:

- Uso de [jackson-jr](#), una biblioteca de serialización que mejora el tiempo de inicialización.
- Uso de las bibliotecas [java.time](#) para los objetos de fecha y hora, parte del JDK.
- Uso de [Slf4j](#) para una fachada de registro.

Recursos adicionales de

La Guía para AWS Lambda desarrolladores contiene una [sección sobre las mejores prácticas](#) para desarrollar funciones de Lambda que no es específica de Java.

Para ver un ejemplo de cómo crear una aplicación nativa de la nube en Java que utilice Java AWS Lambda, consulte el contenido de este [taller](#). En el taller se trata la optimización del rendimiento y otras buenas prácticas.

Puede considerar la posibilidad de utilizar imágenes estáticas que se compilen con antelación para reducir la latencia de startup. Por ejemplo, puede usar el SDK para Java 2.x y Maven para [generar una imagen nativa de GraalVM](#).

Cientes de HTTP

Puede cambiar el cliente HTTP para usarlo en su cliente de servicio, así como cambiar la configuración predeterminada de los clientes HTTP con AWS SDK for Java 2.x. En esta sección se describen los clientes HTTP y la configuración del SDK.

Cientes HTTP disponibles en el SDK para Java

Cientes síncronos

Los clientes HTTP sincrónicos del SDK para Java implementan la [SdkHttpClient](#) interfaz. Un cliente de servicio síncrono, como el `S3Client` o el `DynamoDbClient`, requiere el uso de un cliente HTTP síncrono. AWS SDK for Java Ofrece tres clientes HTTP sincrónicos.

ApacheHttpClient (predeterminado)

[ApacheHttpClient](#) es el cliente HTTP predeterminado para los clientes de servicios síncronos. Para obtener información acerca de cómo configurar el `ApacheHttpClient`, consulte [Configurar el cliente HTTP basado en Apache](#).

AwsCrtHttpClient

[AwsCrtHttpClient](#) proporciona E/S de alto rendimiento y sin bloqueo. Se basa en el cliente HTTP AWS Common Runtime (CRT). Para obtener información sobre cómo configurar el `AwsCrtHttpClient` y utilizar el servicio con los clientes, consulte [the section called “Configurar clientes AWS HTTP basados en CRT”](#).

URLConnectionHttpClient

Para minimizar la cantidad de archivos jar y bibliotecas de terceros que utiliza la aplicación, puede utilizar el [URLConnectionHttpClient](#). Para obtener información acerca de cómo configurar el `URLConnectionHttpClient`, consulte [Configurar el cliente HTTP basado en URLConnection](#).

Clientes asíncronos

Los clientes HTTP asíncronos del SDK for Java implementan la interfaz. [SdkAsyncHttpClient](#) Un cliente de servicio asíncrono, como el `S3AsyncClient` o el `DynamoDbAsyncClient`, requiere el uso de un cliente HTTP asíncrono. AWS SDK for Java Ofrece dos clientes HTTP asíncronos.

`NettyNioAsyncHttpClient` (predeterminado)

[NettyNioAsyncHttpClient](#) es el cliente HTTP predeterminado que utilizan los clientes asíncronos. Para obtener información acerca de cómo configurar el `NettyNioAsyncHttpClient`, consulte [the section called “Configurar el cliente HTTP basado en Netty”](#).

`AwsCrtAsyncHttpClient`

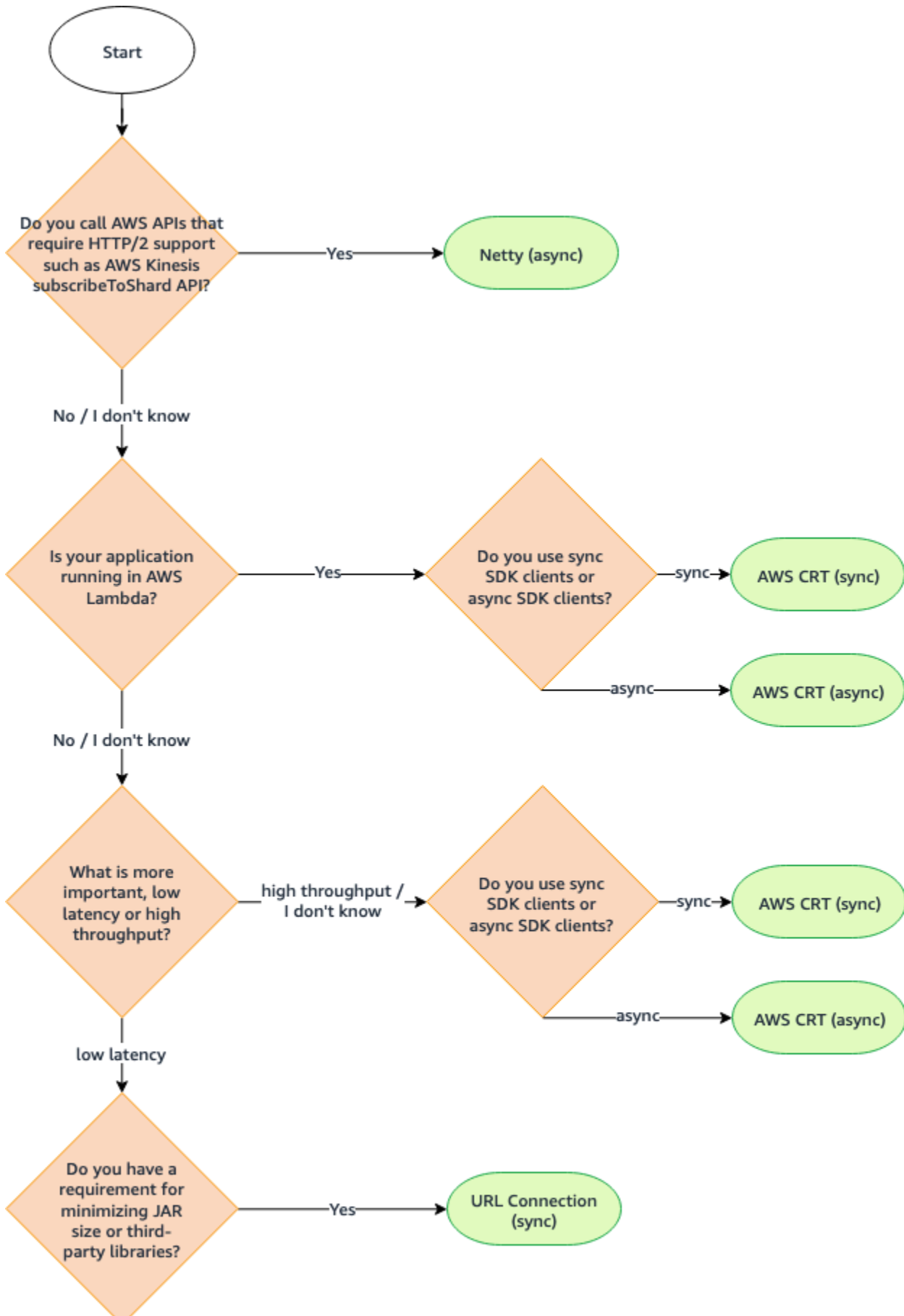
[AwsCrtAsyncHttpClient](#) Se basa en el cliente HTTP AWS Common Runtime (CRT). Para obtener información acerca de cómo configurar el `AwsCrtAsyncHttpClient`, consulte [the section called “Configurar clientes AWS HTTP basados en CRT”](#).

Recomendaciones sobre el cliente HTTP

Al elegir una implementación de cliente HTTP entran en juego varios factores. Utilice la siguiente información para ayudarle en la decisión.

Diagrama de flujo de recomendaciones

El siguiente diagrama de flujo proporciona una guía general para ayudarle a determinar qué cliente HTTP debe utilizar.



Comparación de clientes HTTP

La siguiente tabla proporciona información detallada de cada cliente HTTP.

Cliente HTTP	Sínc. or asínc.	Cuándo se debe usar	Limitación o inconveniente
Cliente HTTP basado en Apache (cliente HTTP de sincronización predeterminado)	Sincronizar	Úselo si prefiere una latencia baja en lugar de un alto rendimiento	Tiempo de startup más lento en comparación con otros clientes HTTP
Cliente HTTP basado en URLConnection	Sincronizar	Úselo si tiene requisitos estrictos para limitar las dependencias de terceros	No admite el método HTTP PATCH, que requieren algunas API, como las operaciones de actualización de Amazon APIGateway
AWS Cliente HTTP de sincronización basado en CRT 1	Sincronizar	<ul style="list-style-type: none"> • Úselo si su aplicación se está ejecutando en AWS Lambda - Utilícelo si prefiere un alto rendimiento a una baja latencia • Úselo si prefiere clientes del SDK síncronos 	N/A
Netty-based HTTP client	Asíncrono	<ul style="list-style-type: none"> • Úselo si su aplicación invoca API que requieren compatibilidad con HTTP/2, como la API de Kinesis SubscribeToShard 	Tiempo de startup más lento en comparación

Cliente HTTP	Sínc. or asínc.	Cuándo se debe usar	Limitación o inconveniente
(cliente HTTP asíncrono predeterminado)			ón con otros clientes HTTP
AWS ^{Cliente HTTP} asíncrono basado en CRT ¹	Asíncrono	<ul style="list-style-type: none"> • Úselo si su aplicación se está ejecutando en AWS Lambda - Utilícelo si prefiere un alto rendimiento a una baja latencia • Úselo si prefiere clientes del SDK asíncronos 	<ul style="list-style-type: none"> • No es compatible con clientes de servicio que requieran compatibilidad con HTTP/2, como KinesisAsyncClient y TranscribeStreamingAsyncClient

¹ Debido a sus ventajas adicionales, le recomendamos que utilice los clientes HTTP basados en AWS CRT si es posible.

Valores predeterminados de configuración inteligente

La AWS SDK for Java 2.x versión 2.17.102 o posterior ofrece una función de configuración inteligente por defecto. Esta característica optimiza dos propiedades del cliente HTTP junto con otras propiedades que no afectan al cliente HTTP.

Los valores predeterminados de la configuración inteligente establecen valores razonables para las propiedades `connectTimeoutInMillis` y `tlsNegotiationTimeoutInMillis` en función del valor de modo predeterminado que proporcione. El valor de modo predeterminado se elige según las características de la aplicación.

Para obtener más información sobre la configuración inteligente predeterminada y sobre cómo elegir el mejor modo predeterminado para sus aplicaciones, consulte la [Guía de referencia de las herramientas y los SDK de AWS](#).

A continuación, se muestran cuatro formas de configurar el modo predeterminado para su aplicación.

Service client

Utilice el generador de clientes de servicio para configurar el modo predeterminado directamente en el cliente de servicio. El siguiente ejemplo establece el modo predeterminado en auto para el `DynamoDbClient`.

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

System property

Puede usar la propiedad del `aws.defaultsMode` sistema para especificar el modo predeterminado. Si establece la propiedad del sistema en Java, debe establecerla antes de inicializar cualquier cliente de servicio.

En el siguiente ejemplo, se muestra cómo configurar el modo predeterminado en auto usar un conjunto de propiedades del sistema en Java.

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

En el siguiente ejemplo, se muestra cómo configurar el modo predeterminado como auto mediante una opción `-D` del comando `java`.

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

Environment variable

Defina un valor para la variable de entorno `AWS_DEFAULTS_MODE` para seleccionar el modo predeterminado de la aplicación.

La siguiente información muestra el comando que se debe ejecutar para establecer el valor del modo predeterminado como auto utilizando una variable de entorno.

Sistema operativo	Comando para definir variables de entorno
Linux, macOS o Unix	<code>export AWS_DEFAULTS_MODE=auto</code>
Windows	<code>set AWS_DEFAULTS_MODE=auto</code>

AWS config file

Puede añadir una propiedad de `defaults_mode` configuración al AWS config archivo compartido, como se muestra en el siguiente ejemplo.

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

Si define el modo predeterminado de forma global con la propiedad del sistema, la variable de entorno o el archivo de configuración AWS , puede sustituir la configuración cuando cree un cliente HTTP.

Al crear un cliente HTTP con el método `httpClientBuilder()`, la configuración se aplica solo a la instancia que se está creando. Un ejemplo de ello se muestra [aquí](#). En este ejemplo, el cliente HTTP basado en Netty anula cualquier valor de modo predeterminado establecido globalmente para `connectTimeoutInMillis` y `tlsNegotiationTimeoutInMillis`.

Configurar el cliente HTTP basado en Apache

Los clientes del servicio sincrónico AWS SDK for Java 2.x utilizan un cliente HTTP basado en Apache de forma predeterminada. [ApacheHttpClient](#) El SDK `ApacheHttpClient` se basa en Apache. [HttpClient](#)

El SDK también ofrece el [URLConnectionHttpClient](#), que se carga más rápido, pero tiene menos funciones. Para obtener información acerca de cómo configurar el `URLConnectionHttpClient`, consulte [the section called “Configurar el cliente HTTP basado en URLConnection”](#).

Para ver el conjunto completo de opciones de configuración disponibles para el `ApacheHttpClient`, consulte [ApacheHttpClient.Builder](#) y [ProxyConfiguration.Builder](#).

Acceso al `ApacheHttpClient`

En la mayoría de las situaciones, `ApacheHttpClient` se utiliza sin ninguna configuración explícita. Declare sus clientes de servicios y el SDK configurará los `ApacheHttpClient` con valores estándar por usted.

Si quiere configurar de forma explícita el `ApacheHttpClient` o usarlo con varios clientes de servicio, tiene que hacerlo disponible para su configuración.

No se necesita configuración

Cuando declara una dependencia de un cliente de servicio en Maven, el SDK añade una dependencia en tiempo de ejecución del artefacto `apache-client`. Esto hace que la clase `ApacheHttpClient` esté disponible para su código en el tiempo de ejecución, pero no en el de compilación. Si no está configurando el cliente HTTP basado en Apache, no necesita especificar una dependencia para él.

En el siguiente fragmento XML de un archivo `pom.xml` Maven, la dependencia declarada con `<artifactId>s3</artifactId>` trae de forma transitoria el cliente HTTP basado en Apache. No necesita declarar una dependencia específicamente para ella.

```
<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>
```

Con estas dependencias, no puede realizar ningún cambio en la configuración HTTP, ya que la biblioteca `ApacheHttpClient` solo se encuentra en la ruta de clases del tiempo de ejecución.

Se necesita configuración

Para configurar el `ApacheHttpClient`, es necesario añadir una dependencia a la biblioteca `apache-client` en el momento de la compilación.

Consulte el siguiente ejemplo de un archivo `pom.xml` Maven para configurar el `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>
```

Utilizar y configurar el `ApacheHttpClient`

Puede configurar una instancia del `ApacheHttpClient` junto con la creación de un cliente de servicio, o configurar una sola instancia para compartirla entre varios clientes de servicio.

Con cualquiera de estos enfoques, utilice [ApacheHttpClient.Builder](#) para configurar las propiedades de la instancia de cliente HTTP basada en Apache.

Práctica recomendada: dedicar una instancia de **ApacheHttpClient** a un cliente de servicio

Si necesita configurar una instancia del `ApacheHttpClient`, le recomendamos que cree la instancia `ApacheHttpClient` dedicada. Puede hacerlo con el método `httpClientBuilder` del generador del cliente del servicio. De esta forma, el SDK administra el ciclo de vida del cliente HTTP, lo que ayuda a evitar posibles pérdidas de memoria si la instancia `ApacheHttpClient` no se cierra cuando ya no se necesita.

En el ejemplo siguiente, se crea un `S3Client` y se configura la instancia integrada de `ApacheHttpClient` con los valores `maxConnections` y `connectionTimeout`. La instancia HTTP se crea mediante el método `httpClientBuilder` de `S3Client.Builder`.

Importaciones

```
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.
```

Enfoque alternativo: compartir una instancia **ApacheHttpClient**

Para reducir el uso de recursos y memoria en su aplicación, puede configurar un `ApacheHttpClient` y compartirlo entre varios clientes de servicio. El grupo de conexiones HTTP se compartirá, lo que reduce el uso de recursos.

Note

Al compartir una instancia de `ApacheHttpClient`, es preciso cerrarla cuando esté lista para ser eliminada. El SDK no cerrará la instancia cuando el cliente del servicio esté cerrado.

En el siguiente ejemplo, se configura un cliente HTTP basado en Apache, utilizado por dos clientes de servicio. La instancia de `ApacheHttpClient` configurada se pasa al método `httpClient` de cada creador. Cuando los clientes de servicio y el cliente HTTP ya no son necesarios, el código los cierra de forma explícita. El código cierra el cliente HTTP por última vez.

Importaciones

```
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.
```

Ejemplo de configuración proxy

El siguiente fragmento de código utiliza el [generador de configuración proxy para el cliente HTTP de Apache](#).

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

Las propiedades del sistema Java equivalentes para la configuración del proxy se muestran en el siguiente fragmento de línea de comandos.

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

La configuración equivalente que usa variables de entorno es:

```
// Set the following environment variables.
// $ export HTTP_PROXY="http://username:password@example.com:1234"
// $ export NO_PROXY="localhost|host.example.com"

// Set the 'useSystemPropertyValues' to false on the proxy configuration.
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .useSystemPropertyValues(Boolean.FALSE)
        .build())
    .build();

// Run the application.
// $ java -cp ... App
```

Note

El cliente HTTP de Apache no admite actualmente las propiedades del sistema proxy HTTPS ni la variable de entorno HTTPS_PROXY.

Configurar el cliente HTTP basado en URLConnection

AWS SDK for Java 2.x Ofrece un cliente [URLConnectionHttpClient](#) HTTP más ligero en comparación con el predeterminado. `URLConnectionHttpClient` se basa en el [URLConnection](#) de Java.

El `URLConnectionHttpClient` se carga más rápido que el cliente HTTP basado en Apache, pero tiene menos características. Como se carga más rápido, es una [buena solución](#) para las funciones AWS Lambda de Java.

El `URLConnectionHttpClient` tiene varias [opciones configurables](#) a las que puede acceder.

Note

El `URLConnectionHttpClient` no admite el método HTTP PATCH.

Algunas operaciones de AWS API requieren solicitudes de PATCH. Los nombres de esas operaciones suelen empezar por `Update*`. A continuación se presentan varios ejemplos.

- [Varias Update* operaciones](#) en la AWS Security Hub API y también en la [BatchUpdateFindings](#) operación
- Todas las [operaciones de Update*](#) de la API de Amazon API Gateway
- [Varias Update* operaciones](#) en la WorkDocs API de Amazon

Si puedes usar el `URLConnectionHttpClient`, consulta primero la referencia de la API Servicio de AWS que estás usando. Compruebe si las operaciones que necesita utilizan la operación PATCH.

Acceso al `URLConnectionHttpClient`

Para configurar y utilizar el `URLConnectionHttpClient`, debe declarar una dependencia en el artefacto Maven `pom.xml` de su archivo `url-connection-client`.

A diferencia del `ApacheHttpClient`, el `URLConnectionHttpClient` se añade automáticamente al proyecto, por lo que el usuario debe declararlo específicamente.

El siguiente ejemplo de un archivo `pom.xml` muestra las dependencias necesarias para usar y configurar el 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>
```

Utilizar y configurar el `URLConnectionHttpClient`

Puede configurar una instancia del `URLConnectionHttpClient` junto con la creación de un cliente de servicio, o configurar una sola instancia para compartirla entre varios clientes de servicio.

Con cualquiera de los dos enfoques, se utiliza el [URLConnectionHttpClient.Builder](#) para configurar las propiedades del cliente HTTP basado en `URLConnection`.

Práctica recomendada: dedicar una instancia de `URLConnectionHttpClient` a un cliente de servicio

Si necesita configurar una instancia del `URLConnectionHttpClient`, le recomendamos que cree la instancia `URLConnectionHttpClient` dedicada. Puede hacerlo con el método `httpClientBuilder` del generador del cliente del servicio. De esta forma, el SDK administra el

ciclo de vida del cliente HTTP, lo que ayuda a evitar posibles pérdidas de memoria si la instancia `URLConnectionHttpClient` no se cierra cuando ya no se necesita.

En el ejemplo siguiente, se crea un `S3Client` y se configura la instancia integrada de `URLConnectionHttpClient` con los valores `socketTimeout` y `proxyConfiguration`. El método `proxyConfiguration` toma una expresión lambda de Java de tipo `Consumer<ProxyConfiguration.Builder>`.

Importaciones

```
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.
```

Enfoque alternativo: compartir una instancia `URLConnectionHttpClient`

Para reducir el uso de recursos y memoria en su aplicación, puede configurar un `URLConnectionHttpClient` y compartirlo entre varios clientes de servicio. El grupo de conexiones HTTP se compartirá, lo que reduce el uso de recursos.

Note

Al compartir una instancia de `URLConnectionHttpClient`, es preciso cerrarla cuando esté lista para ser eliminada. El SDK no cerrará la instancia cuando el cliente del servicio esté cerrado.

En el siguiente ejemplo, se configura un cliente HTTP basado en `URLConnection`, que utilizan dos clientes de servicio. La instancia de `URLConnectionHttpClient` configurada se pasa al método `httpClient` de cada creador. Cuando los clientes de servicio y el cliente HTTP ya no son necesarios, el código los cierra de forma explícita. El código cierra el cliente HTTP por última vez.

Importaciones

```
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();
```

Utiliza `URLConnectionHttpClient` y `ApacheHttpClient` juntos

Al utilizar el `URLConnectionHttpClient` en su aplicación, debe proporcionar a cada cliente de servicio una instancia `URLConnectionHttpClient` o una instancia `ApacheHttpClient` mediante el método `httpClientBuilder` del creador de clientes de servicio.

Se produce una excepción si el programa utiliza varios clientes de servicio y se cumplen las siguientes condiciones:

- Un cliente de servicio está configurado para usar una instancia `URLConnectionHttpClient`
- Otro cliente de servicio utiliza el `ApacheHttpClient` predeterminado sin compilarlo explícitamente con los métodos `httpClient()` o `httpClientBuilder()`

La excepción indicará que se encontraron varias implementaciones HTTP en la ruta de clases.

El siguiente fragmento de código de ejemplo conduce a una excepción.

```
// 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 la excepción configurando explícitamente el `S3Client` con un `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 crear explícitamente el `ApacheHttpClient`, debe [añadir una dependencia](#) del artefacto `apache-client` en su archivo de proyecto Maven.

Ejemplo de configuración proxy

El siguiente fragmento de código utiliza el [generador de configuración proxy para el cliente HTTP de conexión 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();
```

Las propiedades del sistema Java equivalentes para la configuración del proxy se muestran en el siguiente fragmento de línea de comandos.

```
$ java -Dhttp.proxyHost=example.com -Dhttp.proxyPort=1234 -Dhttp.proxyUser=username \
```

```
-Dhttp.proxyPassword=password -Dhttp.nonProxyHosts=localhost|host.example.com -cp ...  
App
```

La configuración equivalente que usa variables de entorno es:

```
// Set the following environment variables.  
// $ export HTTP_PROXY="http://username:password@example.com:1234"  
// $ export NO_PROXY="localhost|host.example.com"  
  
// Set the 'useSystemPropertyValues' to false on the proxy configuration.  
SdkHttpClient apacheHttpClient = UrlConnectionHttpClient.builder()  
    .proxyConfiguration(ProxyConfiguration.builder()  
        .useSystemPropertyValues(Boolean.FALSE)  
        .build())  
    .build();  
  
// Run the application.  
// $ java -cp ... App
```

Note

El cliente HTTP basado en `URLConnection` no admite actualmente las propiedades del sistema proxy HTTPS ni la variable de entorno `HTTPS_PROXY`.

Configurar el cliente HTTP basado en Netty

El cliente HTTP predeterminado para las operaciones asíncronas en el AWS SDK for Java 2.x es el basado en Netty. [NettyNioAsyncHttpClient](#) El cliente basado en Netty se basa en el marco de red asíncrono dirigido por eventos del [proyecto Netty](#).

Como cliente HTTP alternativo, puede utilizar el nuevo [cliente HTTP basado en CRT de AWS](#). En este tema se muestra cómo configurar el `NettyNioAsyncHttpClient`.

Acceso al `NettyNioAsyncHttpClient`

En la mayoría de las situaciones, se utiliza `NettyNioAsyncHttpClient` sin ninguna configuración explícita en los programas asíncronos. Declara sus clientes de servicios asíncronos y el SDK configurará los `NettyNioAsyncHttpClient` con valores estándar por usted.

Si quiere configurar de forma explícita el `NettyNioAsyncHttpClient` o usarlo con varios clientes de servicio, tiene que hacerlo disponible para su configuración.

No se necesita configuración

Cuando declara una dependencia de un cliente de servicio en Maven, el SDK añade una dependencia en tiempo de ejecución del artefacto `netty-nio-client`. Esto hace que la clase `NettyNioAsyncHttpClient` esté disponible para su código en el tiempo de ejecución, pero no en el de compilación. Si no está configurando el cliente HTTP basado en Netty, no necesita especificar una dependencia para él.

En el siguiente fragmento XML de un archivo `pom.xml` Maven, la dependencia declarada con `<artifactId>dynamodb-enhanced</artifactId>` trae de forma transitoria el cliente HTTP basado en Netty. No necesita declarar una dependencia específicamente para ella.

```
<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>
```

Con estas dependencias, no puede realizar ningún cambio en la configuración HTTP, ya que la biblioteca `NettyNioAsyncHttpClient` solo se encuentra en la ruta de clases del tiempo de ejecución.

Se necesita configuración

Para configurar el `NettyNioAsyncHttpClient`, es necesario añadir una dependencia al artefacto `netty-nio-client` en tiempo de compilación.

Consulte el siguiente ejemplo de un archivo `pom.xml` Maven para configurar el `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>
```

Utilizar y configurar el `NettyNioAsyncHttpClient`

Puede configurar una instancia del `NettyNioAsyncHttpClient` junto con la creación de un cliente de servicio, o configurar una sola instancia para compartirla entre varios clientes de servicio.

Con cualquiera de los dos enfoques, se utiliza el [NettyNioAsyncHttpClient.Builder](#) para configurar las propiedades de la instancia de cliente HTTP basada en Netty.

Práctica recomendada: dedicar una instancia de `NettyNioAsyncHttpClient` a un cliente de servicio

Si necesita configurar una instancia del `NettyNioAsyncHttpClient`, le recomendamos que cree una instancia de `NettyNioAsyncHttpClient` dedicada. Puede hacerlo con el método `httpClientBuilder` del generador del cliente del servicio. De esta forma, el SDK administra el

ciclo de vida del cliente HTTP, lo que ayuda a evitar posibles pérdidas de memoria si la instancia `NettyNioAsyncHttpClient` no se cierra cuando ya no se necesita.

En el siguiente ejemplo, se crea una instancia `DynamoDbAsyncClient`, que también es utilizada por una instancia `DynamoDbEnhancedAsyncClient`. La instancia `DynamoDbAsyncClient` contiene la instancia `NettyNioAsyncHttpClient` con los valores `connectionTimeout` y `maxConcurrency`. La instancia HTTP se crea mediante el método `httpClientBuilder` de `DynamoDbAsyncClient.Builder`.

Importaciones

```
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();
```

Enfoque alternativo: compartir una instancia **NettyNioAsyncHttpClient**

Para reducir el uso de recursos y memoria en su aplicación, puede configurar un `NettyNioAsyncHttpClient` y compartirlo entre varios clientes de servicio. El grupo de conexiones HTTP se compartirá, lo que reduce el uso de recursos.

Note

Al compartir una instancia de `NettyNioAsyncHttpClient`, es preciso cerrarla cuando esté lista para ser eliminada. El SDK no cerrará la instancia cuando el cliente del servicio esté cerrado.

En el siguiente ejemplo, se configura un cliente HTTP basado en Netty, utilizado por dos clientes de servicio. La instancia de `NettyNioAsyncHttpClient` configurada se pasa al método `httpClient` de cada creador. Cuando los clientes de servicio y el cliente HTTP ya no son necesarios, el código los cierra de forma explícita. El código cierra el cliente HTTP por última vez.

Importaciones

```
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.
```

Ejemplo de configuración proxy

El siguiente fragmento de código utiliza el [generador de configuración proxy para el cliente HTTP de Netty](#).

```
SdkAsyncHttpClient nettyHttpClient = NettyNioAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .nonProxyHosts(Set.of("localhost", "host.example.com"))
        .build())
    .build();
```

Las propiedades del sistema Java equivalentes para la configuración del proxy se muestran en el siguiente fragmento de línea de comandos.

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

Important

Para utilizar cualquiera de las propiedades del sistema proxy HTTPS, la propiedad `scheme` debe estar configurada en código para `https`. Si la propiedad del esquema no está configurada en el código, el esquema predeterminado es HTTP y el SDK solo busca las propiedades del sistema `http.*`.

La configuración equivalente que usa variables de entorno es:

```
// Set the following environment variables.
// $ export HTTPS_PROXY="https://username:password@myproxy:1234"
// $ export NO_PROXY="localhost|host.example.com"

// Set the 'useSystemPropertyValues' to false on the proxy configuration.
SdkAsyncHttpClient nettyHttpClient = NettyNioAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .useSystemPropertyValues(Boolean.FALSE)
        .build())
    .build();

// Run the application.
// $ java -cp ... App
```

Configurar clientes AWS HTTP basados en CRT

Los clientes HTTP AWS basados en CRT incluyen los clientes síncronos y asíncronos.

[AwsCrtHttpClient](#)[AwsCrtAsyncHttpClient](#) Los clientes HTTP AWS basados en CRT ofrecen las siguientes ventajas para los clientes HTTP:

- Tiempo de inicio del SDK más rápido
- Ocupación de menos espacio de memoria
- Tiempo de latencia reducido
- Administración del estado de conexión

- equilibrio de carga de DNS

AWS Componentes basados en CRT en el SDK

Los clientes HTTP AWS basados en CRT, que se describen en este tema, y el cliente S3 AWS basado en CRT son componentes diferentes del SDK.

Los clientes HTTP basados en CRT de AWS síncronos y asíncronos son implementaciones e interfaces de cliente HTTP de SDK y se utilizan para la comunicación HTTP general. Son alternativas a los otros clientes HTTP síncronos o asíncronos del SDK con ventajas adicionales.

El [cliente S3 AWS basado en CRT](#) es una implementación de la AsyncClient interfaz [S3](#) y se utiliza para trabajar con el servicio Amazon S3. Es una alternativa a la implementación de la interfaz `S3AsyncClient` basada en Java y ofrece varias ventajas.

Si bien ambos componentes utilizan bibliotecas del [AWS Common Runtime](#), los clientes HTTP AWS basados en CRT no utilizan la [biblioteca aws-c-s 3](#) y no admiten las funciones de la API de carga [multiparte de S3](#). El cliente S3 AWS basado en CRT, por el contrario, se creó específicamente para admitir las funciones de la API de carga multiparte de S3.

Acceda a los clientes HTTP basados en CRT AWS

Antes de poder usar los clientes HTTP AWS basados en CRT, agrega el `aws-crt-client` artefacto con una versión mínima de 2.22.0 a las dependencias de tu proyecto.

La siguiente guía Maven `pom.xml` muestra los clientes HTTP AWS basados en CRT declarados mediante el mecanismo de lista de materiales (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>
</project>
```

```
</dependency>
</dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>aws-crt-client</artifactId>
  </dependency>
</dependencies>
</project>
```

Visite el repositorio central de Maven para ver [la versión más reciente](#).

Utilice y configure un cliente HTTP basado en CRT AWS

Puede configurar un cliente HTTP AWS basado en CRT junto con la creación de un cliente de servicio, o puede configurar una sola instancia para compartirla entre varios clientes de servicio.

Con cualquiera de estos enfoques, se utiliza un generador para [configurar las propiedades de la instancia](#) de cliente HTTP AWS basada en CRT.

Práctica recomendada: dedicar una instancia de a un cliente de servicio

Si necesitas configurar una instancia de un cliente HTTP AWS basado en CRT, te recomendamos que dediques la instancia y la compiles junto con el cliente de servicio. Puede hacerlo con el método `httpClientBuilder` del generador del cliente del servicio. De esta forma, el SDK administra el ciclo de vida del cliente HTTP, lo que ayuda a evitar posibles pérdidas de memoria si la instancia del cliente HTTP AWS basada en CRT no se cierra cuando ya no es necesaria.

En el siguiente ejemplo, se crea un cliente de servicio S3 y se configura un cliente HTTP AWS basado en CRT con valores y valores. `connectionTimeout` `maxConcurrency`

Synchronous client

Importaciones

```
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

Importaciones

```
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();
```

Enfoque alternativo: compartir una instancia

Para reducir el uso de recursos y memoria para su aplicación, puede configurar un cliente HTTP AWS basado en CRT y compartirlo entre varios clientes de servicio. El grupo de conexiones HTTP se compartirá, lo que reduce el uso de recursos.

Note

Cuando se comparte una instancia de cliente HTTP AWS basada en CRT, debe cerrarla cuando esté lista para su eliminación. El SDK no cerrará la instancia cuando el cliente del servicio esté cerrado.

El siguiente ejemplo configura una instancia de cliente HTTP AWS basada en CRT con valores y `connectionTimeout` `maxConcurrency` La instancia configurada se pasa al método `httpClient` del creador de cada cliente de servicio. Cuando los clientes de servicio y el cliente HTTP ya no son necesarios, se cierran de forma explícita. El cliente HTTP se cierra en último lugar.

Synchronous client**Importaciones**

```
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

Importaciones

```
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.
```

Establezca un cliente HTTP AWS basado en CRT como predeterminado

Puede configurar su archivo de compilación de Maven para que el SDK utilice un cliente HTTP AWS basado en CRT como cliente HTTP predeterminado para los clientes de servicio.

Para ello, añada un elemento `exclusions` con las dependencias predeterminadas del cliente HTTP a cada artefacto del cliente de servicio.

En el siguiente `pom.xml` ejemplo, el SDK usa un cliente HTTP AWS basado en CRT para los servicios de S3. Si el cliente de servicio de su código es un `S3AsyncClient`, el SDK utiliza `AwsCrtAsyncHttpClient`. Si el cliente de servicio es un `S3Client`, el SDK utiliza `AwsCrtHttpClient`. Con esta configuración, el cliente HTTP asíncrono predeterminado basado en Netty y el HTTP síncrono predeterminado basado en Apache no están disponibles.

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

```
        <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 el repositorio central de Maven para ver el valor de la [*VERSIÓN más reciente*](#).

Note

Si se declaran varios clientes de servicio en un archivo pom.xml, todos requieren el elemento XML exclusions.

Utilizar una propiedad del sistema Java

Para usar los clientes HTTP AWS basados en CRT como HTTP predeterminado para su aplicación, puede establecer la propiedad `software.amazon.awssdk.http.async.service.impl` del sistema Java en un valor de `software.amazon.awssdk.http.crt.AwsCrtSdkHttpService`

Para configurarlo durante el inicio de la aplicación, ejecute un comando similar al siguiente.

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

Use el siguiente fragmento de código para establecer la propiedad del sistema en el código de la aplicación.

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

Note

Debe añadir una dependencia al `aws-crt-client` artefacto del `pom.xml` archivo cuando utilice una propiedad del sistema para configurar el uso de los clientes HTTP basados en AWS CRT.

Configuración avanzada de clientes HTTP basados en CRT de AWS

Puede utilizar varios ajustes de configuración de los clientes HTTP AWS basados en CRT, como la configuración del estado de la conexión y el tiempo máximo de inactividad. Puede revisar las [opciones de configuración disponibles](#) para el `AwsCrtAsyncHttpClient`. Puede configurar las mismas opciones para `AwsCrtHttpClient`.

Configuración del estado de conexión

Puede configurar el estado de la conexión para los clientes HTTP AWS basados en CRT mediante el `connectionHealthConfiguration` método del generador de clientes HTTP.

En el siguiente ejemplo, se crea un cliente de servicio S3 que utiliza una instancia de cliente HTTP AWS basada en CRT configurada con una configuración de mantenimiento de la conexión y un tiempo máximo de inactividad para las conexiones.

Synchronous client

Importaciones

```
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

Importaciones

```
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();
```

Compatibilidad con HTTP/2

El protocolo HTTP/2 aún no es compatible con los clientes HTTP AWS basados en CRT, pero está previsto que se publique en un futuro.

Mientras tanto, si utiliza clientes de servicio que requieren compatibilidad con HTTP/2, como el [KinesisAsyncClient](#) o el [TranscribeStreamingAsyncClient](#), considere la posibilidad de utilizar el en su lugar. [NettyNioAsyncHttpClient](#)

Ejemplo de configuración proxy

El siguiente fragmento de código muestra el uso de [ProxyConfiguration.Builder](#) que se utiliza para configurar el proxy en el código.

Synchronous client

Importaciones

```
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")
        .nonProxyHosts(Set.of("localhost", "host.example.com")))
        .build())
    .build();
```

Asynchronous client

Importaciones

```
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")
        .nonProxyHosts(Set.of("localhost", "host.example.com"))
        .build()
    .build();

```

Las propiedades del sistema Java equivalentes para la configuración del proxy se muestran en el siguiente fragmento de línea de comandos.

```

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

```

Important

Para utilizar cualquiera de las propiedades del sistema proxy HTTPS, la propiedad `scheme` debe estar configurada en código para `https`. Si la propiedad del esquema no está configurada en el código, el esquema predeterminado es `HTTP` y el SDK solo busca las propiedades del sistema `http.*`.

La configuración equivalente que usa variables de entorno es:

```

// Set the following environment variables.
// $ export HTTPS_PROXY="https://username:password@myproxy:1234"
// $ export NO_PROXY="localhost|host.example.com"

// Set the 'useSystemPropertyValues' to false on the proxy configuration.
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .useSystemPropertyValues(Boolean.FALSE)
        .build())
    .build();

// Run the application.
// $ java -cp ... App

```

Configurar proxies HTTP

Puede configurar los proxies HTTP mediante código, estableciendo las propiedades del sistema Java o configurando variables de entorno.

Configure en código

Los proxies se configuran en código con un generador `ProxyConfiguration` específico del cliente al crear el cliente de servicio. El código siguiente muestra un ejemplo de configuración de proxy para un cliente HTTP basado en Apache que utiliza un cliente de servicio Amazon S3.

```
SdkHttpClient httpClient1 = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://proxy.example.com"))
        .username("username")
        .password("password")
        .addNonProxyHost("localhost")
        .build())
    .build();

S3Client s3Client = S3Client.builder()
    .httpClient(httpClient)
    .build();
```

En la sección correspondiente a cada cliente HTTP de este tema se muestra un ejemplo de configuración de proxy.

- [Cliente HTTP Apache](#)
- [URLConnection-based HTTP client](#)
- [Netty-based HTTP client](#)
- [AWS Cliente HTTP basado en CRT](#)

Configure los proxies HTTP con ajustes externos

Aunque no utilices explícitamente un `ProxyConfiguration` compilador en el código, el SDK busca ajustes externos para configurar una configuración de proxy predeterminada.

De forma predeterminada, el SDK busca primero las propiedades del sistema JVM. Si se encuentra una sola propiedad, el SDK usa el valor y cualquier otro valor de propiedad del sistema. Si no hay propiedades del sistema disponibles, el SDK busca variables de entorno proxy.

El SDK puede usar las siguientes propiedades del sistema Java y variables de entorno.

Propiedades del sistema Java

Propiedad del sistema	Descripción	Compatibilidad con clientes HTTP
http.proxyHost	Nombre de host del servidor proxy HTTP	Todos
http.proxyPort	Número de puerto del servidor proxy HTTP	Todos
http.proxyUser	Nombre de usuario para la autenticación mediante proxy HTTP	Todos
http.proxyPassword	Contraseña para la autenticación mediante proxy HTTP	Todos
http.nonProxyHosts	Lista de hosts a los que se debe acceder directamente, sin pasar por el proxy. Esta lista también es válida cuando se utiliza HTTPS.	Todos
https.proxyHost	Nombre de host del servidor proxy HTTP	Netty, CRT
https.proxyPort	Número de puerto del servidor proxy HTTP	Netty, CRT
https.proxyUser	Nombre de usuario para la autenticación mediante proxy HTTPS	Netty, CRT
https.proxyPassword	Contraseña para la autenticación mediante proxy HTTPS	Netty, CRT

Variables de entorno

Variable de entorno	Descripción	Compatibilidad con clientes HTTP
HTTP_PROXY ¹	Una URL válida con un esquema de HTTP	Todos
HTTPS_PROXY ¹	Una URL válida con un esquema de HTTPS	Netty, CRT
NO_PROXY ²	Lista de hosts a los que se debe acceder directamente, sin pasar por el proxy. La lista es válida tanto para HTTP como para HTTPS.	Todos

Ver notas clave y a pie de página

Todos: todos los clientes HTTP que ofrece el SDK: `URLConnectionHttpClient`, `ApacheHttpClientNettyNioAsyncHttpClient`, `AwsCrtAsyncHttpClient`.

Netty: el cliente HTTP basado en Netty (`NettyNioAsyncHttpClient`)

CRT: los clientes HTTP AWS basados en CRT, (`AwsCrtHttpClient` y `AwsCrtAsyncHttpClient`)

¹ La variable de entorno consultada, ya sea `HTTP_PROXY` o `HTTPS_PROXY`, depende de la configuración del esquema del cliente. `ProxyConfiguration` El esquema predeterminado es HTTP. El siguiente fragmento muestra cómo cambiar el esquema a HTTPS utilizado para la resolución de variables de entorno.

```
SdkHttpClient httpClient = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .build())
    .build();
```

² La variable de `NO_PROXY` entorno admite una combinación de separadores «|» y «,» entre los nombres de host. Los nombres de host pueden incluir el comodín «*».

Utilice una combinación de ajustes

Puede usar una combinación de configuraciones de proxy HTTP en el código, las propiedades del sistema y las variables de entorno.

Example — configuración proporcionada por una propiedad del sistema y por código

```
// Command line with the proxy password set as a system property.
$ java -Dhttp.proxyPassword=SYS_PROP_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();
```

El SDK resuelve la siguiente configuración de proxy.

```
Host = localhost
Port = 1234
Password = SYS_PROP_password
UserName = username
Non ProxyHost = null
```

Example — están disponibles tanto las propiedades del sistema como las variables de entorno

El `ProxyConfiguration` creador de cada cliente HTTP ofrece configuraciones denominadas `useSystemPropertyValues` y `useEnvironmentVariablesValues`. De forma predeterminada, ambas configuraciones son `true`. Cuando `true`, el SDK utiliza automáticamente los valores de las propiedades del sistema o las variables de entorno para las opciones que no proporciona el `ProxyConfiguration` generador.

⚠ Important

Las propiedades del sistema tienen prioridad sobre las variables de entorno. Si se encuentra una propiedad del sistema proxy HTTP, el SDK recupera todos los valores de las propiedades del sistema y ninguno de las variables de entorno. Si desea priorizar las variables de entorno por encima de las propiedades del sistema, use `useSystemPropertyValues` establézcalas en `false`

Para este ejemplo, los siguientes ajustes están disponibles en tiempo de ejecución:

```
// System properties
http.proxyHost=SYS_PROP_HOST.com
http.proxyPort=2222
http.password=SYS_PROP_PASSWORD
http.user=SYS_PROP_USER

// Environment variables
HTTP_PROXY="http://EnvironmentUser:EnvironmentPassword@ENV_VAR_HOST:3333"
NO_PROXY="environmentnonproxy.host,environmentnonproxy2.host:1234"
```

El cliente de servicio se crea con una de las siguientes instrucciones. Ninguna de las sentencias establece explícitamente una configuración de proxy.

```
DynamoDbClient client = DynamoDbClient.create();
DynamoDbClient client = DynamoDbClient.builder().build();
DynamoDbClient client = DynamoDbClient.builder()
    .httpClient(ApacheHttpClient.builder()
        .proxyConfiguration(ProxyConfiguration.builder()
            .build())
        .build())
    .build();
```

El SDK resuelve las siguientes configuraciones de proxy:

```
Host = SYS_PROP_HOST.com
Port = 2222
Password = SYS_PROP_PASSWORD
UserName = SYS_PROP_USER
Non ProxyHost = null
```

Como el cliente de servicio tiene la configuración de proxy predeterminada, el SDK busca las propiedades del sistema y, a continuación, las variables de entorno. Como la configuración de las propiedades del sistema tiene prioridad sobre las variables de entorno, el SDK utiliza únicamente las propiedades del sistema.

Si el uso de las propiedades del sistema se cambia al `false` que se muestra en el código siguiente, el SDK resuelve solo las variables de entorno.

```
DynamoDbClient client = DynamoDbClient.builder()
    .httpClient(ApacheHttpClient.builder()
        .proxyConfiguration(ProxyConfiguration.builder()
            .useSystemPropertyValues(Boolean.FALSE)
            .build())
        .build())
    .build();
```

La configuración de proxy resuelta mediante HTTP es:

```
Host = ENV_VAR_HOST
Port = 3333
Password = EnvironmentPassword
UserName = EnvironmentUser
Non ProxyHost = environmentnonproxy.host, environmentnonproxy2.host:1234
```

Tratamiento de excepciones para el AWS SDK for Java 2.x

Saber cómo y cuándo AWS SDK for Java 2.x produce excepciones es importante para crear aplicaciones de alta calidad con el SDK. En las siguientes secciones se describen los diferentes casos de excepciones que produce el SDK y cómo tratarlas correctamente.

¿Por qué excepciones no controladas?

AWS SDK for Java utiliza excepciones en tiempo de ejecución (o no controladas) en lugar de excepciones controladas por los siguientes motivos:

- Para permitir a los desarrolladores un control minucioso de los errores que desean administrar sin obligarles a abordar casos excepcionales que no les preocupan (o que les obligan a detallar su código en exceso)
- Para evitar problemas de escalabilidad inherentes a las excepciones controladas en aplicaciones grandes

En general, las excepciones controladas funcionan bien a pequeña escala, pero pueden ser problemáticas cuando las aplicaciones crecen y se vuelven más complejas.

AwsServiceException (y subclases)

[AwsServiceException](#) es la excepción más común que experimentará al AWS SDK for Java usar. `AwsServiceException` es una subclase de las más generales [SdkServiceException](#). `AwsServiceException` representan una respuesta de error de un Servicio de AWS. Por ejemplo, si intenta terminar una instancia Amazon EC2 que no existe, Amazon EC2 devolverá una respuesta de error y todos los detalles de dicha respuesta de error se incluirán en la excepción `AwsServiceException` que se produce.

Cuando encuentre una `AwsServiceException`, sabrá que la solicitud se ha enviado correctamente al Servicio de AWS, pero que no se ha podido procesar correctamente. Esto puede ser debido a errores en los parámetros de la solicitud o a problemas en el servicio.

`AwsServiceException` proporciona información como:

- Código de estado HTTP devuelto
- Código de error de AWS devuelto
- Mensaje de error detallado del servicio de la [AwsErrorDetails](#) clase
- ID de solicitud de AWS de la solicitud que ha producido un error

En algunos casos, se produce una subclase de `AwsServiceException` para permitir a los desarrolladores un control minucioso del tratamiento de casos de error a través de bloques de captura. La referencia de la API del SDK de Java [AwsServiceException](#) muestra la gran cantidad de `AwsServiceException` subclases. Utilice los enlaces de las subclases para profundizar y ver las excepciones granulares generadas por un servicio.

Por ejemplo, los siguientes enlaces a la referencia API del SDK muestran las jerarquías de excepciones para algunos Servicios de AWS comunes. La lista de subclases que se muestra en cada página muestra las excepciones específicas que su código puede capturar.

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

Para obtener más información sobre una excepción, inspeccione `errorCode` el [AwsErrorDetails](#) objeto. Puede usar el valor `errorCode` para buscar información en la API de la guía de servicios. Por ejemplo, si se captura un `S3Exception` y el valor `AwsErrorDetails#errorCode()` es `InvalidRequest`, utilice la [lista de códigos de error](#) de la Referencia de la API de Amazon S3 para ver más detalles.

SdkClientException

[SdkClientException](#) indica que se ha producido un problema en el código del cliente de Java, ya sea al intentar enviar una solicitud AWS o al analizar una respuesta desde él AWS. Una `SdkClientException` es, por lo general, más grave que una `SdkServiceException`, e indica un problema importante que impide que el cliente haga llamadas de servicio a los servicios de AWS. Por ejemplo, AWS SDK for Java produce una excepción `SdkClientException` si no está disponible la conexión de red cuando intenta llamar a una operación en uno de los clientes.

Excepciones y comportamiento de reintentos

El SDK para Java vuelve a intentar las solicitudes de varias [excepciones del cliente](#) y de los [códigos de estado HTTP](#) que recibe de las respuestas Servicio de AWS. Estos errores se gestionan como parte del `RetryMode` heredado que los clientes de servicios utilizan de forma predeterminada. La referencia de la API de Java para [RetryMode](#) describe las distintas formas de configurar el modo.

Para personalizar las excepciones y los códigos de estado HTTP que activan los reintentos automáticos, configure su cliente de servicio con una [RetryPolicy](#) que añada instancias [RetryOnExceptionsCondition](#) y [RetryOnStatusCodeCondition](#).

Reintentos

Las llamadas a Servicios de AWS pueden fallar ocasionalmente por motivos inesperados. Algunos errores, como la limitación (se ha superado la frecuencia) o los errores transitorios, pueden producirse si se vuelve a intentar realizar la llamada. AWS SDK for Java 2.x Cuenta con un mecanismo integrado para detectar estos errores y reintentar automáticamente la llamada, que está activado de forma predeterminada para todos los clientes.

En esta página se describe cómo funciona, cómo configurar los distintos modos y personalizar el comportamiento de los reintentos.

Estrategias de reintento

Una estrategia de reintentos es un mecanismo que se utiliza en el SDK para implementar los reintentos. Cada cliente del SDK tiene una estrategia de reintentos creada en el momento de la compilación que no se puede modificar una vez creado el cliente.

La estrategia de reintentos tiene las siguientes responsabilidades.

- Clasifique las excepciones como reintentables o no.
- Calcule el retraso sugerido para esperar antes del siguiente intento.
- Mantenga un [depósito de fichas](#) que proporcione un mecanismo para detener los reintentos cuando un gran porcentaje de las solicitudes fallen y los reintentos no tengan éxito.

Note

Antes del lanzamiento de las estrategias de reintento con la versión 2.26.0 del SDK, las políticas de reintentos proporcionaban el mecanismo de reintento en el SDK. La API de políticas de reintentos se compone de la [RetryPolicy](#) clase principal del `software.amazon.awssdk.core.retry` paquete, mientras que el [software.amazon.awssdk.retries](#) paquete contiene los elementos de la API de la estrategia de reintentos.

La API de estrategia de reintentos se introdujo como parte de un esfuerzo AWS general por unificar las interfaces y el comportamiento de los componentes principales de los SDK.

El SDK para Java 2.x tiene tres estrategias de reintento integradas: estándar, heredada y adaptativa. Las tres estrategias de reintento están preconfiguradas para reintentarlo en un conjunto de excepciones que se pueden volver a intentar. Algunos ejemplos de errores que se pueden volver a intentar son los tiempos de espera de los sockets, la limitación del lado del servicio, los fallos de bloqueo simultáneos o optimistas y los errores de servicio transitorios.

Estrategia de reintentos estándar

La [estrategia de reintento estándar](#) es la `RetryStrategy` implementación recomendada para los casos de uso normales. A diferencia de esta `AdaptiveRetryStrategy`, la estrategia estándar suele ser útil en todos los casos de uso de reintentos.

De forma predeterminada, la estrategia de reintentos estándar hace lo siguiente.

- Se vuelve a intentar en las condiciones configuradas en el momento de la compilación. Puede ajustar esto con `StandardRetryStrategy.Builder#retryOnException`.
- Lo vuelve a intentar 2 veces para un total de 3 intentos. Puedes ajustar esto con `StandardRetryStrategy.Builder#maxAttempts(int)`.
- Utiliza la estrategia de `BackoffStrategy#exponentialDelay` retroceso, con un retraso base de 100 milisegundos y un retraso máximo de 20 segundos. Puedes ajustarlo con `StandardRetryStrategy.Builder#backoffStrategy`.
- Interrumpe el circuito (deshabilita los reintentos) en caso de que se produzcan grandes fallos en la fase descendente. El primer intento siempre se ejecuta, solo los reintentos están deshabilitados. Ajústelo con `StandardRetryStrategy.Builder#circuitBreakerEnabled`.

Estrategia de reintento tradicional

La [estrategia de reintento tradicional](#) es `RetryStrategy` para casos de uso normales, sin embargo, ha quedado obsoleta en favor de la `StandardRetryStrategy`. Esta es la estrategia de reintentos predeterminada que utilizan los clientes cuando no especifican otra estrategia.

Se caracteriza por tratar las excepciones con y sin regulación de manera diferente: en el caso de las excepciones con regulación, el retraso base del retardo es mayor (500 ms) que el retraso base para las excepciones sin limitación (100 ms), y las excepciones con limitación no afectan al estado del depósito de fichas.

La experiencia en el uso de esta estrategia a gran escala ha demostrado que no es particularmente mejor que la estrategia de reintento estándar. Además, AWS no protege los servicios intermedios de las tormentas de reintentos y puede provocar una escasez de recursos por parte del cliente.

De forma predeterminada, la estrategia de reintentos tradicional hace lo siguiente.

- Se vuelve a intentar en las condiciones configuradas en el momento de la compilación. Puede ajustar esto con `LegacyRetryStrategy.Builder#retryOnException`.
- Lo vuelve a intentar 3 veces para un total de 4 intentos. Puedes ajustar esto con `LegacyRetryStrategy.Builder#maxAttempts(int)`.
- En el caso de excepciones no limitantes, utiliza la estrategia de `BackoffStrategy#exponentialDelay` retroceso, con un retraso base de 100 milisegundos y un retraso máximo de 20 segundos. Puedes ajustarlo con `RetryStrategy.Builder#backoffStrategy`.

- Para las excepciones de limitación, utiliza la estrategia de `BackoffStrategy#exponentialDelay` retroceso, con un retraso base de 500 milisegundos y un retraso máximo de 20 segundos. Puedes ajustar esto con `LegacyRetryStrategy.Builder#throttlingBackoffStrategy`
- Interrumpe el circuito (deshabilita los reintentos) en caso de que se produzcan grandes fallos en la fase descendente. La interrupción del circuito nunca impide un primer intento exitoso. Puede ajustar este comportamiento con `LegacyRetryStrategy.Builder#circuitBreakerEnabled`.
- El estado del disyuntor no se ve afectado por las excepciones de regulación.

Estrategia de reintento adaptativa

La [estrategia de reintento adaptativo](#) es `RetryStrategy` para casos de uso con un alto nivel de limitaciones de recursos.

La estrategia de reintentos adaptativa incluye todas las características de la estrategia estándar y añade un limitador de velocidad por parte del cliente que mide la tasa de solicitudes restringidas en comparación con las solicitudes no restringidas. La estrategia utiliza esta medida para ralentizar las solicitudes e intentar mantenerlas dentro de un ancho de banda seguro. Lo ideal es que no se produzcan errores de limitación.

De forma predeterminada, la estrategia de reintento adaptativo hace lo siguiente.

- Se vuelve a intentar en las condiciones configuradas en el momento de la compilación. Puede ajustar esto con `AdaptiveRetryStrategy.Builder#retryOnException`.
- Lo vuelve a intentar 2 veces para un total de 3 intentos. Puedes ajustar esto con `AdaptiveRetryStrategy.Builder#maxAttempts(int)`.
- Utiliza un retardo de retroceso dinámico que se basa en la carga actual contra el recurso descendente.
- Interrumpe el circuito (deshabilita los reintentos) cuando hay un gran número de fallos en la fase descendente. La interrupción del circuito puede impedir un segundo intento en situaciones de interrupción para proteger el servicio descendente.

⚠ Warning

La estrategia de reintento adaptativo supone que el cliente trabaja con un único recurso (por ejemplo, una tabla de DynamoDB o un bucket de Amazon S3).

Si utiliza un solo cliente para varios recursos, la limitación o las interrupciones asociadas a un recurso provocan un aumento de la latencia y errores cuando el cliente accede a todos los demás recursos. Cuando utilice la estrategia de reintento adaptativo, le recomendamos que utilice un único cliente para cada recurso.

También le recomendamos que utilice esta estrategia en situaciones en las que todos los clientes utilicen la estrategia de reintento adaptativo contra el recurso.

⚠ Important

La publicación de las estrategias de reintento con la versión 2.26.0 del SDK de Java incluye el nuevo valor de enumeración. [RetryMode.ADAPTIVE_V2](#) El ADAPTIVE_V2 modo corrige un error que no pudo retrasar el primer intento cuando se detectaron errores de limitación anteriormente.

Con la versión 2.26.0, los usuarios obtienen automáticamente el comportamiento del ADAPTIVE_V2 modo configurándolo como `adaptive` si fuera una variable de entorno, una propiedad del sistema o un ajuste de perfil. No hay ningún `adaptive_v2` valor para estos ajustes. Consulte la siguiente [the section called “Especifique una estrategia”](#) sección para saber cómo configurar el modo.

Los usuarios pueden obtener el comportamiento anterior configurando el modo en el código mediante `RetryMode.ADAPTIVE`.

Resumen: Comparación de los valores predeterminados de la estrategia de reintento

En la siguiente tabla se muestran los valores predeterminados de las propiedades de cada estrategia de reintento.

Strategy (Estrategia)	Número máximo de intentos	Retraso base para errores no relacionados con la regulación	Retardo base para los errores de regulación	Tamaño del bucket de tokens	Coste simbólico por reintento sin limitación	Coste simbólico por reintento de limitación
Estándar	3	100	100	500	5	5
Legacy	4	100	500	500	5	0
Adaptativo	3	100	100	500	5	5

Especifique una estrategia

Dispone de cuatro formas de especificar una estrategia para su cliente de servicio.

En el código

Al crear un cliente, puede configurar una expresión lambda con una estrategia de reintento. El siguiente fragmento configura una estrategia de reintento estándar que utiliza valores predeterminados en un cliente de servicio de DynamoDB.

```
DynamoDbClient client = DynamoDbClient.builder()
    .overrideConfiguration(o -> o.retryStrategy(RetryMode.STANDARD))
    .build();
```

Puede especificar o en lugar de. `RetryMode.LEGACY` `RetryMode.ADAPTIVE` `RetryMode.STANDARD`

Como configuración de perfil

Incluir `retry_mode` como configuración de perfil en el [archivo de AWS configuración compartido](#). Especifique `standardlegacy`, o `adaptive` como un valor. Cuando se establece como configuración de perfil, todos los clientes de servicio que se crean mientras el perfil está activo utilizan la estrategia de reintento especificada con los valores predeterminados. Puede anular esta configuración configurando una estrategia de reintentos en el código, como se muestra anteriormente.

Con el siguiente perfil, todos los clientes del servicio utilizan la estrategia de reintentos estándar.

```
[profile dev]
region = us-east-2
retry_mode = standard
```

Como propiedad del sistema JVM

Puede configurar una estrategia de reintentos para todos los clientes del servicio, a menos que se anule en el código, mediante la propiedad del sistema. `aws.retryMode` Especificue `standard`, o `adaptive` como `legacy` un valor.

Use el `-D` modificador cuando invoque Java, como se muestra en el siguiente comando.

```
java -Daws.retryMode=standard ...
```

Como alternativa, defina la propiedad del sistema en el código antes de crear cualquier cliente, tal y como se muestra en el siguiente fragmento.

```
public void main(String[] args) {
    // Set the property BEFORE any AWS service clients are created.
    System.setProperty("aws.retryMode", "standard");
    ...
}
```

Con una variable de entorno

También puede utilizar la variable de `AWS_RETRY_MODE` entorno con un valor de `standard` `legacy`, o `adaptive`. Al igual que ocurre con una configuración de perfil o una propiedad del sistema JVM, la variable de entorno configura todos los clientes de servicio con el modo de reintento especificado, a menos que se configure un cliente en código.

El siguiente comando establece el modo de reintento `standard` para la sesión de shell actual.

```
export AWS_RETRY_MODE=standard
```

Personaliza una estrategia

Puedes personalizar cualquier estrategia de reintentos estableciendo el número máximo de intentos, la estrategia de retroceso y las excepciones que se pueden volver a intentar. Puede personalizar

el momento de crear una estrategia de reintentos o de crear un cliente mediante un generador de anulaciones que permita seguir perfeccionando la estrategia configurada.

Personaliza el número máximo de intentos

Puede configurar el número máximo de intentos durante la construcción del cliente, como se muestra en la siguiente declaración. La siguiente sentencia personaliza la estrategia de reintentos predeterminada para el cliente hasta un máximo de 5 intentos: un primer intento más 4 reintentos.

```
DynamoDbClient client = DynamoDbClient.builder()
    .overrideConfiguration(o -> o.retryStrategy(b -> b.maxAttempts(5)))
    .build();
```

Como alternativa, puede crear la estrategia y proporcionársela al cliente, tal y como se muestra en el siguiente ejemplo de código. El siguiente código reemplaza el máximo de 3 intentos estándar por 10 y configura un cliente DynamoDB con la estrategia personalizada.

```
StandardRetryStrategy strategy = AwsRetryStrategy.standardRetryStrategy()
    .toBuilder()
    .maxAttempts(10)
    .build();
DynamoDbClient client = DynamoDbClient.builder()
    .overrideConfiguration(o -> o.retryStrategy(strategy))
    .build();
```

Warning

Se recomienda configurar cada cliente con una instancia única. `RetryStrategy` Si se comparte una `RetryStrategy` instancia, los errores de un cliente pueden afectar al comportamiento de reintento del otro.

También puedes establecer el número máximo de intentos para todos los clientes mediante [ajustes externos](#) en lugar de código. Este ajuste se configura como se describe en la [the section called “Especifique una estrategia”](#) sección.

Personalice las excepciones que se pueden volver a intentar

Puede configurar excepciones adicionales que provoquen retiros durante la construcción del cliente. Esta personalización se proporciona para casos extremos en los que se lanzan excepciones que no están incluidas en el conjunto predeterminado de excepciones que se pueden volver a intentar.

En el siguiente fragmento de código se muestran los métodos que se utilizan para personalizar las excepciones de reintento: `y.retryOnException` `retryOnExceptionOrCause` El `retryOnExceptionOrCause` método agrega una excepción que se puede volver a intentar si el SDK lanza la excepción directa o si la excepción está empaquetada.

```
DynamoDbClient client = DynamoDbClient.builder()
    .overrideConfiguration(o -> o.retryStrategy(
        b -> b.retryOnException(EdgeCaseException.class)
            .retryOnExceptionOrCause(WrappedEdgeCaseException.class)))
    .build();
```

Personaliza la estrategia de retroceso

Puede crear la estrategia de reducción y proporcionársela al cliente.

El siguiente código crea una `BackoffStrategy` que reemplaza la estrategia de retardo exponencial de la estrategia estándar predeterminada.

```
BackoffStrategy backoffStrategy =
    BackoffStrategy.exponentialDelay(Duration.ofMillis(150), // The base delay.
        Duration.ofSeconds(15)); // The maximum delay.
DynamoDbClient client = DynamoDbClient.builder()
    .overrideConfiguration(o -> o.retryStrategy(
        b -> b.backoffStrategy(backoffStrategy)))
    .build();
```

Migración desde **RetryPolicy** a **RetryStrategy**

`RetryPolicy` (la API de política de reintentos) será compatible en un futuro próximo. Si actualmente utilizas una instancia de `RetryPolicy` para configurar tu cliente, todo va a funcionar como antes. Entre bastidores, el SDK de Java lo adapta a un `RetryStrategy`. Las nuevas interfaces de estrategia de reintento proporcionan la misma funcionalidad que una, `RetryPolicy` pero se crean y configuran de forma diferente.

Inicio de sesión con SDK para Java 2.x

AWS SDK for Java 2.x Utiliza [SLF4J](#), que es una capa de abstracción que permite el uso de cualquiera de los varios sistemas de registro en tiempo de ejecución.

Los sistemas de registro compatibles incluyen Java Logging Framework y Apache [Log4j 2](#), entre otros. En este tema se muestra cómo utilizar Log4j 2 como sistema de registro para trabajar con el SDK.

Archivo de configuración de Log4j 2

Por lo general, se utiliza un archivo de configuración, cuyo nombre es `log4j2.xml` con Log4j 2. A continuación, se muestran ejemplos de archivos de configuración. Para obtener más información acerca de los valores que se utilizan en el archivo de configuración, consulte el [manual de configuración de Log4j](#).

El archivo `log4j2.xml` debe estar en la ruta de clases cuando se inicie la aplicación. Para un proyecto de Maven, coloque el archivo en el directorio `<project-dir>/src/main/resources`.

El archivo de configuración `log4j2.xml` especifica propiedades como el [nivel de registro](#), dónde se envía la salida de registro (por ejemplo, [a un archivo o a la consola](#)) y el [formato de la salida](#). El nivel de registro especifica el nivel de detalle que genera Log4j 2. Log4j 2 admite el concepto de varias [jerarquías](#) de registro. El nivel de registro se define de forma independiente para cada jerarquía. La jerarquía de registro principal que se utiliza con el es. `software.amazon.awssdk`

Añadir dependencia de registro

Para configurar el enlace Log4j 2 para SLF4J en su archivo de compilación, utilice lo siguiente.

Maven

Añada el siguiente elemento a su archivo `pom.xml`.

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


Gradle–Kotlin DSL

Añada lo siguiente a su archivo `build.gradle.kts`.

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

Utilice `2.20.0` para la versión mínima del artefacto `log4j-slf4j2-impl`. Para obtener la última versión, utilice la versión publicada en [Maven Central](#). Sustituya `VERSION` por la que vaya a utilizar.

Errores y advertencias específicos del SDK

Le recomendamos que deje siempre la jerarquía del registrador `software.amazon.awssdk` establecida en `WARN` para captar cualquier mensaje importante de las bibliotecas cliente del SDK. Por ejemplo, si el cliente Amazon S3 detecta que su aplicación no ha cerrado correctamente un `InputStream` y podría estar desperdiciando recursos, el cliente S3 informa de ello a través de un mensaje de advertencia a los archivos log. Esto también garantiza que se registren los mensajes si el cliente tiene algún problema con el tratamiento de las solicitudes o respuestas.

El siguiente archivo `log4j2.xml` establece `rootLogger` en `«WARN»`, lo que genera mensajes de advertencia y de nivel de error de todos los registradores de la aplicación, incluidos los de la jerarquía `«software.amazon.awssdk»`. Como alternativa, puede establecer explícitamente la jerarquía del registrador `software.amazon.awssdk` en `WARN` si se utiliza `<Root level="ERROR">`.

Ejemplo de archivo de configuración Log4j2.xml

Esta configuración registrará los mensajes en los niveles `«ERROR»` y `«WARN»` en la consola para todas las jerarquías 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 de resumen de solicitud/respuesta

Cada solicitud a un Servicio de AWS genera un identificador de AWS solicitud único que resulta útil si tienes algún problema con la forma en Servicio de AWS que maneja una solicitud. AWS Se puede acceder a los ID de solicitud mediante programación a través de [SdkServiceException](#) los objetos del SDK para cualquier llamada de servicio fallida, y también se pueden informar a través del nivel de registro «DEBUG» del registrador «software.amazon.awssdk.request».

El siguiente archivo `log4j2.xml` habilita un resumen de solicitudes y respuestas.

```
<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>
```

Este es un ejemplo del resultado del registro:

```
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Sending Request:
  DefaultSdkHttpFullRequest(httpMethod=POST, protocol=https, host=dynamodb.us-
  east-1.amazonaws.com, encodedPath=/, headers=[amz-sdk-invocation-id, Content-Length,
  Content-Type, User-Agent, X-Amz-Target], queryParameters=[])
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Received
  successful response: 200, Request ID:
```

```
QS9DUMME2NHEDH8TGT9N5V530JVV4KQNS05AEMVJF66Q9ASUAAJG, Extended Request ID: not available
```

Si solo le interesa el identificador de solicitud, utilice `<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />`.

Registro del SDK a nivel de depuración

Si necesitas más detalles sobre lo que hace el SDK, puedes configurar el nivel de registro del `software.amazon.awssdk` registrador en `DEBUG`. En este nivel, el SDK genera una gran cantidad de detalles, por lo que te recomendamos que establezcas este nivel para resolver los errores mediante pruebas de integración.

En este nivel de registro, el SDK registra información sobre la configuración, la resolución de credenciales, los interceptores de ejecución, la actividad de TLS de alto nivel, la firma de solicitudes y mucho más.

La siguiente es una muestra de las sentencias que genera el SDK a `DEBUG` nivel de una `S3Client#listBuckets()` llamada.

```
DEBUG s.a.a.r.p.AwsRegionProviderChain:57 - Unable to load region from
software.amazon.awssdk.regions.providers.SystemSettingsRegionProvider@324dcd31:Unable
to load region from system settings. Region must be specified either via environment
variable (AWS_REGION) or system property (aws.region).
DEBUG s.a.a.c.i.h.l.ClasspathSdkHttpServiceProvider:85 - The HTTP implementation loaded
is software.amazon.awssdk.http.apache.ApacheSdkHttpService@a23a01d
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Creating an interceptor
chain that will apply interceptors in the following order:
[software.amazon.awssdk.core.internal.interceptor.HttpChecksumValidationInterceptor@69b2f8e5,
software.amazon.awssdk.awscore.interceptor.HelpfulUnknownHostExceptionInterceptor@6331250e,
software.amazon.awssdk.awscore.eventstream.EventStreamInitialRequestInterceptor@a10c1b5,
software.amazon.awssdk.awscore.interceptor.TraceIdExecutionInterceptor@644abb8f,
software.amazon.awssdk.services.s3.auth.scheme.internal.S3AuthSchemeInterceptor@1a411233,
software.amazon.awssdk.services.s3.endpoints.internal.S3ResolveEndpointInterceptor@70325d20,
software.amazon.awssdk.services.s3.endpoints.internal.S3RequestSetEndpointInterceptor@7c2327fa,
software.amazon.awssdk.services.s3.internal.handlers.StreamingRequestInterceptor@4d847d32,
software.amazon.awssdk.services.s3.internal.handlers.CreateBucketInterceptor@5f462e3b,
software.amazon.awssdk.services.s3.internal.handlers.CreateMultipartUploadRequestInterceptor@3,
software.amazon.awssdk.services.s3.internal.handlers.DecodeUrlEncodedResponseInterceptor@58065,
software.amazon.awssdk.services.s3.internal.handlers.GetBucketPolicyInterceptor@3605c4d3,
software.amazon.awssdk.services.s3.internal.handlers.S3ExpressChecksumInterceptor@585c13de,
software.amazon.awssdk.services.s3.internal.handlers.AsyncChecksumValidationInterceptor@187eb9
```

```
software.amazon.awssdk.services.s3.internal.handlers.SyncChecksumValidationInterceptor@726a6b9
software.amazon.awssdk.services.s3.internal.handlers.EnableTrailingChecksumInterceptor@6ad11a5
software.amazon.awssdk.services.s3.internal.handlers.ExceptionTranslationInterceptor@522b2631,
software.amazon.awssdk.services.s3.internal.handlers.GetObjectInterceptor@3ff57625,
software.amazon.awssdk.services.s3.internal.handlers.CopySourceInterceptor@1ee29c84,
software.amazon.awssdk.services.s3.internal.handlers.ObjectMetadataInterceptor@7c8326a4]
DEBUG s.a.a.u.c.CachedSupplier:85 - (SsoOidcTokenProvider()) Cached value is stale and
will be refreshed.
...
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Creating an interceptor
chain that will apply interceptors in the following order:
[software.amazon.awssdk.core.internal.interceptor.HttpChecksumValidationInterceptor@51351f28,
software.amazon.awssdk.awscore.interceptor.HelpfulUnknownHostExceptionInterceptor@21618fa7,
software.amazon.awssdk.awscore.eventstream.EventStreamInitialRequestInterceptor@15f2eda3,
software.amazon.awssdk.awscore.interceptor.TraceIdExecutionInterceptor@34cf294c,
software.amazon.awssdk.services.sso.auth.scheme.internal.SsoAuthSchemeInterceptor@4d7aaca2,
software.amazon.awssdk.services.sso.endpoints.internal.SsoResolveEndpointInterceptor@604b1e1d,
software.amazon.awssdk.services.sso.endpoints.internal.SsoRequestSetEndpointInterceptor@625668
...
DEBUG s.a.a.request:85 - Sending Request: DefaultSdkHttpFullRequest(httpMethod=GET,
protocol=https, host=portal.sso.us-east-1.amazonaws.com, encodedPath=/federation/
credentials, headers=[amz-sdk-invocation-id, User-Agent, x-amz-sso_bearer_token],
queryParameters=[role_name, account_id])
DEBUG s.a.a.c.i.h.p.s.SigningStage:85 - Using SelectedAuthScheme: smithy.api#noAuth
DEBUG s.a.a.h.a.i.c.SdkTlsSocketFactory:366 - Connecting socket to portal.sso.us-
east-1.amazonaws.com/18.235.195.183:443 with timeout 2000
...
DEBUG s.a.a.requestId:85 - Received successful response: 200, Request ID: bb4f40f4-
e920-4b5c-8648-58f26e7e08cd, Extended Request ID: not available
DEBUG s.a.a.request:85 - Received successful response: 200, Request ID: bb4f40f4-
e920-4b5c-8648-58f26e7e08cd, Extended Request ID: not available
DEBUG s.a.a.u.c.CachedSupplier:85 -
  (software.amazon.awssdk.services.sso.auth.SsoCredentialsProvider@b965857) Successfully
  refreshed cached value. Next Prefetch Time: 2024-04-25T22:03:10.097Z. Next Stale Time:
  2024-04-25T22:05:30Z
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Interceptor
'software.amazon.awssdk.services.s3.endpoints.internal.S3RequestSetEndpointInterceptor@7c2327f
modified the message with its modifyHttpRequest method.
...
DEBUG s.a.a.c.i.h.p.s.SigningStage:85 - Using SelectedAuthScheme: aws.auth#sigv4
...
DEBUG s.a.a.a.s.Aws4Signer:85 - AWS4 Canonical Request: GET
...
DEBUG s.a.a.h.a.a.i.s.DefaultV4RequestSigner:85 - AWS4 String to sign: AWS4-HMAC-SHA256
```

```
20240425T210631Z
20240425/us-east-1/s3/aws4_request
aafb7784627fa7a49584256cb746279751c48c2076f813259ef767ecce304d64
DEBUG s.a.a.h.a.i.c.SdkTlsSocketFactory:366 - Connecting socket to s3.us-
east-1.amazonaws.com/52.217.41.86:443 with timeout 2000
...
```

El siguiente `log4j2.xml` archivo configura la salida anterior.

```
<Configuration status="WARN">
  <Appenders>
    <Console name="ConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%-5p %c{1.}:%L - %m%n" />
    </Console>
  </Appenders>

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

Habilite el registro de cables

Puede ser útil ver exactamente las solicitudes y respuestas que el SDK para Java 2.x envía y recibe. Si necesita acceder a esta información, puede habilitarla temporalmente añadiendo la configuración necesaria en función del cliente HTTP que utilice el cliente del servicio.

De forma predeterminada, los clientes de servicios síncronos, como el [S3Client](#), utilizan un Apache HttpClient subyacente y los clientes de servicios asíncronos, como el [S3 AsyncClient](#), utilizan un cliente HTTP Netty sin bloqueo.

Este es un desglose de los clientes HTTP que puede utilizar para las dos categorías de clientes de servicio:

Cientes HTTP síncronos	Cientes HTTP asíncronos
ApacheHttpClient (predeterminado)	NettyNioAsyncHttpClient (predeterminado)

Cientes HTTP síncronos	Cientes HTTP asíncronos
URLConnectionHttpClient	AwsCrtAsyncHttpClient

Consulte la pestaña correspondiente a continuación para ver los ajustes de configuración que necesite añadir en función del cliente HTTP subyacente.

Warning

Le recomendamos que solo utilice en registro en red para fines de depuración. Deshabilítelo en sus entornos de producción, ya que puede registrar información confidencial. Registra la solicitud o respuesta completa sin cifrado, incluso para una llamada HTTPS. En el caso de solicitudes o respuestas de gran tamaño (por ejemplo, para cargar un archivo Amazon S3), el registro detallado de las conexiones también puede afectar considerablemente al rendimiento de la aplicación.

ApacheHttpClient

Añada el registrador «org.apache.http.wire» al archivo de configuración `log4j2.xml` y establezca el nivel en «DEBUG».

El siguiente `log4j2.xml` archivo activa el registro completo de cables para 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>
```

Se requiere una dependencia adicional de Maven en el artefacto `log4j-1.2-api` para el registro de cables con Apache, ya que utiliza 1.2 internamente.

El conjunto completo de dependencias de Maven para `log4j 2`, incluido el registro de conexiones para el cliente HTTP Apache, se muestra en los siguientes fragmentos de archivos de compilación.

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 de 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")
}
...
```

Utilice `2.20.0` para la versión mínima del artefacto `log4j-bom`. Para obtener la última versión, utilice la versión publicada en [Maven Central](#). Sustituya **VERSIÓN** por la que vaya a utilizar.

URLConnectionHttpClient

Para registrar los detalles de los clientes del servicio que utilizan la `URLConnectionHttpClient`, cree primero un archivo `logging.properties` con el siguiente contenido:

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

Defina la siguiente propiedad del sistema JVM con la ruta completa de `logging.properties`:

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

Esta configuración registrará únicamente los encabezados de la solicitud y la respuesta, por ejemplo:

```
<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 los cuerpos de la solicitud/respuesta, agregue `-Djavax.net.debug=all` a las propiedades de la JVM. Esta propiedad adicional registra una gran cantidad de información, incluida toda la información de SSL.

En la consola de registro o en el archivo de registro, busque "GET" o "POST" para ir rápidamente a la sección del registro que contiene las solicitudes y respuestas reales. Busque "Plaintext before ENCRYPTION" para las solicitudes y "Plaintext after DECRYPTION" para las respuestas para ver el texto completo de las cabeceras y los contenidos.

NettyNioAsyncHttpClient

Si su cliente de servicio asíncrono utiliza el `NettyNioAsyncHttpClient` predeterminado, añada dos registradores adicionales al su archivo `log4j2.xml` para registrar los encabezados HTTP y los textos de las solicitudes o respuestas.

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

Este es un ejemplo completo de `log4j2.xml`:

```
<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>
```

Esta configuración registra todos los detalles del encabezado y los contenidos de la solicitud/respuesta.

AwsCrtAsyncHttpClient

Si ha configurado su cliente de servicio para que utilice una instancia de `AwsCrtAsyncHttpClient`, puede registrar los detalles configurando las propiedades del sistema JVM o mediante programación.

Log to a file at "Debug" level

Uso de las propiedades del sistema:

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

Mediante programación:

```
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

Uso de las propiedades del sistema:

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

Mediante programación:

```
import software.amazon.awssdk.crt.Log;

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

Por motivos de seguridad, en el nivel de «Rastreo», el `AwsCrtAsyncHttpClient` solo registra los encabezados de respuesta. Los encabezados de las solicitudes, los textos de las solicitudes y los de respuesta no se registran.

Configurar el TTL de JVM para las búsquedas de nombres DNS

La máquina virtual de Java (JVM) almacena en caché las búsquedas de nombres DNS. Cuando la JVM resuelve un nombre de host en una dirección IP, almacena en caché la dirección IP durante un período de tiempo específico, conocido como TTL time-to-live.

Como AWS los recursos utilizan entradas de nombres DNS que cambian de vez en cuando, le recomendamos que configure la JVM con un valor TTL de 5 segundos. Con esto se asegurará de que cuando cambie la dirección IP de un recurso, su aplicación pueda recibir y utilizar la nueva dirección IP del recurso volviendo a consultar el DNS.

En algunas configuraciones de Java, el TTL predeterminado de JVM está establecido de forma que nunca se actualicen las entradas DNS hasta que se reinicie la JVM. Por lo tanto, si la dirección IP de un AWS recurso cambia mientras la aplicación aún se está ejecutando, no podrá usar ese recurso hasta que reinicie manualmente la JVM y se actualice la información de IP almacenada en caché. En este caso, es fundamental establecer el TTL de la JVM de forma que actualice periódicamente la información de las direcciones IP almacenada en caché.

Cómo configurar el TTL de JVM

Para modificar el TTL de la JVM, defina el valor de la propiedad de seguridad [networkaddress.cache.ttl](#) y establezca la propiedad en el archivo para Java 8 o en el `$JAVA_HOME/jre/lib/security/java.security` archivo para `networkaddress.cache.ttl` Java 11 o superior. `$JAVA_HOME/conf/security/java.security`

El siguiente es un fragmento de un archivo que muestra la caché TTL configurada en 5 segundos `java.security`.

```
#
# This is the "master security properties file".
#
# An alternate java.security properties file may be specified
...
# The Java-level namelookup cache policy for successful lookups:
#
# any negative value: caching forever
# any positive value: the number of seconds to cache an address for
# zero: do not cache
...
networkaddress.cache.ttl=5
...
```

Todas las aplicaciones que se ejecutan en la JVM representada por la variable de `$JAVA_HOME` entorno utilizan esta configuración.

Prácticas recomendadas para AWS SDK for Java 2.x

En esta sección, se describen las prácticas recomendadas para utilizar el SDK para Java 2.x.

Temas

- [Reutilizar un cliente del SDK, si es posible](#)
- [Cerrar las secuencias de entrada de las operaciones del cliente](#)
- [Ajustar las configuraciones HTTP en función de las pruebas de rendimiento](#)
- [Utilizar OpenSSL para el cliente HTTP basado en Netty](#)
- [Configurar los tiempos de espera de la API](#)
- [Utilizar métricas](#)

Reutilizar un cliente del SDK, si es posible

Cada cliente del SDK mantiene su propio grupo de conexiones HTTP. Una conexión que ya existe en el grupo se puede reutilizar mediante una nueva solicitud para reducir el tiempo necesario para establecer una nueva conexión. Recomendamos compartir una sola instancia del cliente para evitar la sobrecarga que supone tener demasiados grupos de conexiones que no se utilizan de forma eficaz. Todos los clientes del SDK son seguros para subprocessos.

Si no quiere compartir una instancia de cliente, utilice `close()` en la instancia para liberar los recursos cuando el cliente no sea necesario.

Cerrar las secuencias de entrada de las operaciones del cliente

Para operaciones de secuencia, como [S3Client#getObject](#), si trabaja directamente con [ResponseInputStream](#), recomendamos que haga lo siguiente:

- Lea todos los datos de la secuencia de entrada lo antes posible.
- Cierre la secuencia de entrada cuanto antes.

Hacemos estas recomendaciones porque la secuencia de entrada es un flujo directo de datos de la conexión HTTP y la conexión HTTP subyacente no se puede reutilizar hasta que se hayan leído todos los datos de la secuencia y esta se cierre. Si no se siguen estas reglas, el cliente puede quedarse sin recursos por asignar demasiadas conexiones HTTP abiertas pero no utilizadas.

Ajustar las configuraciones HTTP en función de las pruebas de rendimiento

El SDK proporciona un conjunto de [configuraciones http predeterminadas](#) que se aplican a los casos de uso generales. Recomendamos a los clientes que definan las configuraciones HTTP de sus aplicaciones en función de sus casos de uso.

Como buen punto de partida, el SDK ofrece una característica de [configuración inteligente de valores predeterminados](#). Esta característica está disponible a partir de la versión 2.17.102. Debe seleccionar un modo según su uso, lo que proporciona valores de configuración razonables.

Utilizar OpenSSL para el cliente HTTP basado en Netty

De forma predeterminada, el [NettyNioAsyncHttpClient](#) de SDK utiliza la implementación SSL predeterminada del JDK como `SslProvider`. Nuestras pruebas revelaron que OpenSSL funciona mejor que la implementación predeterminada de JDK. La comunidad de Netty también [recomienda usar OpenSSL](#).

Para usar OpenSSL, agregue `netty-tcnative` a sus dependencias. Para obtener detalles de configuración, consulte la [documentación del proyecto Netty](#).

Una vez que haya configurado su proyecto con `netty-tcnative`, la instancia del `NettyNioAsyncHttpClient` seleccionará OpenSSL automáticamente. Como alternativa, puede configurar el `SslProvider` de forma explícita mediante el compilador de `NettyNioAsyncHttpClient`, como muestra el siguiente fragmento.

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

Configurar los tiempos de espera de la API

El SDK proporciona [valores predeterminados](#) para algunas opciones de tiempo de espera, como el tiempo de espera de la conexión y los tiempos de espera del socket, pero no para los tiempos de espera de las llamadas a la API ni para los tiempos de espera de los intentos de llamadas individuales a la API. Se recomienda establecer tiempos de espera tanto para los intentos individuales como para toda la solicitud. Esto garantizará que la aplicación pueda responder rápido a los errores y de forma óptima cuando se produzcan problemas transitorios que puedan provocar que los intentos de solicitud tarden más en completarse o surjan problemas graves de red.

Puede configurar los tiempos de espera para todas las solicitudes realizadas por los clientes de un servicio mediante [ClientOverrideConfiguration#apiCallAttemptTimeout](#) y [ClientOverrideConfiguration#apiCallTimeout](#).

En el siguiente ejemplo, se muestra la configuración de un cliente Amazon S3 con valores de tiempo de espera personalizados.

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

apiCallAttemptTimeout

Esta configuración establece la cantidad de tiempo para un único intento HTTP, tras el cual se puede volver a intentar la llamada a la API.

apiCallTimeout

El valor de esta propiedad configura la cantidad de tiempo de toda la ejecución, incluidos todos los reintentos.

Como alternativa a establecer estos valores de tiempo de espera en el cliente de servicio, puede usar [RequestOverrideConfiguration#apiCallTimeout\(\)](#) y [RequestOverrideConfiguration#apiCallAttemptTimeout\(\)](#) para configurar una sola solicitud.

El ejemplo siguiente configura una única solicitud de `listBuckets` con valores de tiempo de espera personalizados.

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

Al utilizar juntas estas propiedades, establece un límite estricto en el tiempo total dedicado a todos los intentos entre reintentos. También configura una solicitud HTTP individual para poder responder rápido a los errores en las solicitudes lentas.

Utilizar métricas

El SDK para Java puede [recopilar métricas](#) para los clientes de servicio de su aplicación. Puede utilizar estas métricas para identificar problemas de rendimiento, revisar las tendencias generales de uso, revisar las excepciones de clientes de servicio devueltas o profundizar para comprender un problema concreto.

Le recomendamos que recopile métricas y, a continuación, analice los registros de Amazon CloudWatch para conocer mejor el rendimiento de su aplicación.

Preguntas frecuentes de solución de problemas

A medida que lo utilice AWS SDK for Java 2.x en sus aplicaciones, es posible que se produzcan los errores de tiempo de ejecución que se enumeran en este tema. Utilice las sugerencias que aparecen aquí para ayudarle a descubrir la causa raíz y resolver el error.

¿Cómo puedo solucionar el error «**java.net.SocketException: Se restableció la conexión**» o «el servidor no pudo completar la respuesta»?

Un error al restablecer la conexión indica que el anfitrión Servicio de AWS, el servidor o cualquier intermediario (por ejemplo, una puerta de enlace NAT, un proxy o un equilibrador de carga) cerró la conexión antes de que se completara la solicitud. Como las causas son múltiples, para encontrar una solución es necesario saber por qué se cierra la conexión. Los siguientes elementos suelen provocar el cierre de una conexión.

- La conexión está inactiva. Esto es habitual en las operaciones de streaming, en las que los datos no se escriben ni salen de la red durante un período de tiempo, por lo que un intermediario detecta que la conexión está interrumpida y la cierra. Para evitarlo, asegúrate de que tu aplicación descargue o cargue datos de forma activa.
- Has cerrado el cliente HTTP o SDK. Asegúrese de no cerrar los recursos mientras estén en uso.
- Un proxy mal configurado. Intenta omitir los proxies que hayas configurado para comprobar si eso soluciona el problema. Si esto soluciona el problema, el proxy está cerrando tu conexión por alguna razón. Investiga tu proxy específico para determinar por qué cierra la conexión.

Si no puede identificar el problema, intente ejecutar un volcado de TCP para una conexión afectada en el extremo del cliente de la red (por ejemplo, después de cualquier servidor proxy que controle).

Si ve que el AWS dispositivo de punto final está enviando un TCP RST restablecimiento, [póngase en contacto con el servicio afectado](#) para ver si pueden determinar el motivo del restablecimiento. Prepárate para proporcionar los identificadores de solicitud y las marcas horarias que indiquen cuándo se produjo el problema. El equipo de AWS soporte también podría beneficiarse de [los registros de cableado](#) que muestren exactamente qué bytes envía y recibe su aplicación y cuándo.

¿Cómo puedo corregir el «tiempo de espera de la conexión»?

Un error en el tiempo de espera de la conexión indica que el host Servicio de AWS, el servidor o cualquier intermediario (por ejemplo, una puerta de enlace NAT, un proxy o un equilibrador de carga) no pudieron establecer una nueva conexión con el servidor dentro del tiempo de espera de conexión configurado. En los siguientes elementos se describen las causas más comunes de este problema.

- El tiempo de espera de la conexión configurado es demasiado bajo. De forma predeterminada, el tiempo de espera de la conexión es de 2 segundos en AWS SDK for Java 2.x Si estableces el tiempo de espera de la conexión demasiado bajo, es posible que aparezca este error. El tiempo de espera de conexión recomendado es de 1 segundo si solo haces llamadas dentro de una región y de 3 segundos si realizas solicitudes entre regiones.
- Un proxy mal configurado. Intente omitir los proxies que haya configurado para comprobar si esto soluciona el problema. Si esto soluciona el problema, el proxy es el motivo del tiempo de espera de la conexión. Investiga tu proxy específico para determinar por qué sucede eso

Si no puede identificar el problema, intente ejecutar un volcado de TCP para una conexión afectada en el extremo del cliente de la red (por ejemplo, después de cualquier servidor proxy que controle) para investigar cualquier problema de red.

¿Cómo puedo solucionar el problema

«`java.net.SocketTimeoutException`: Se ha agotado el tiempo de espera de lectura»?

Un error de lectura agotada indica que la JVM intentó leer los datos del sistema operativo subyacente, pero los datos no se devolvieron dentro del tiempo configurado mediante el SDK. Este error puede producirse si el sistema operativo Servicio de AWS, la empresa o cualquier intermediario (por ejemplo, una puerta de enlace NAT, un proxy o un equilibrador de carga) no envía los datos en el tiempo esperado por la JVM. Como las causas son múltiples, para encontrar una solución es necesario saber por qué no se devuelven los datos.

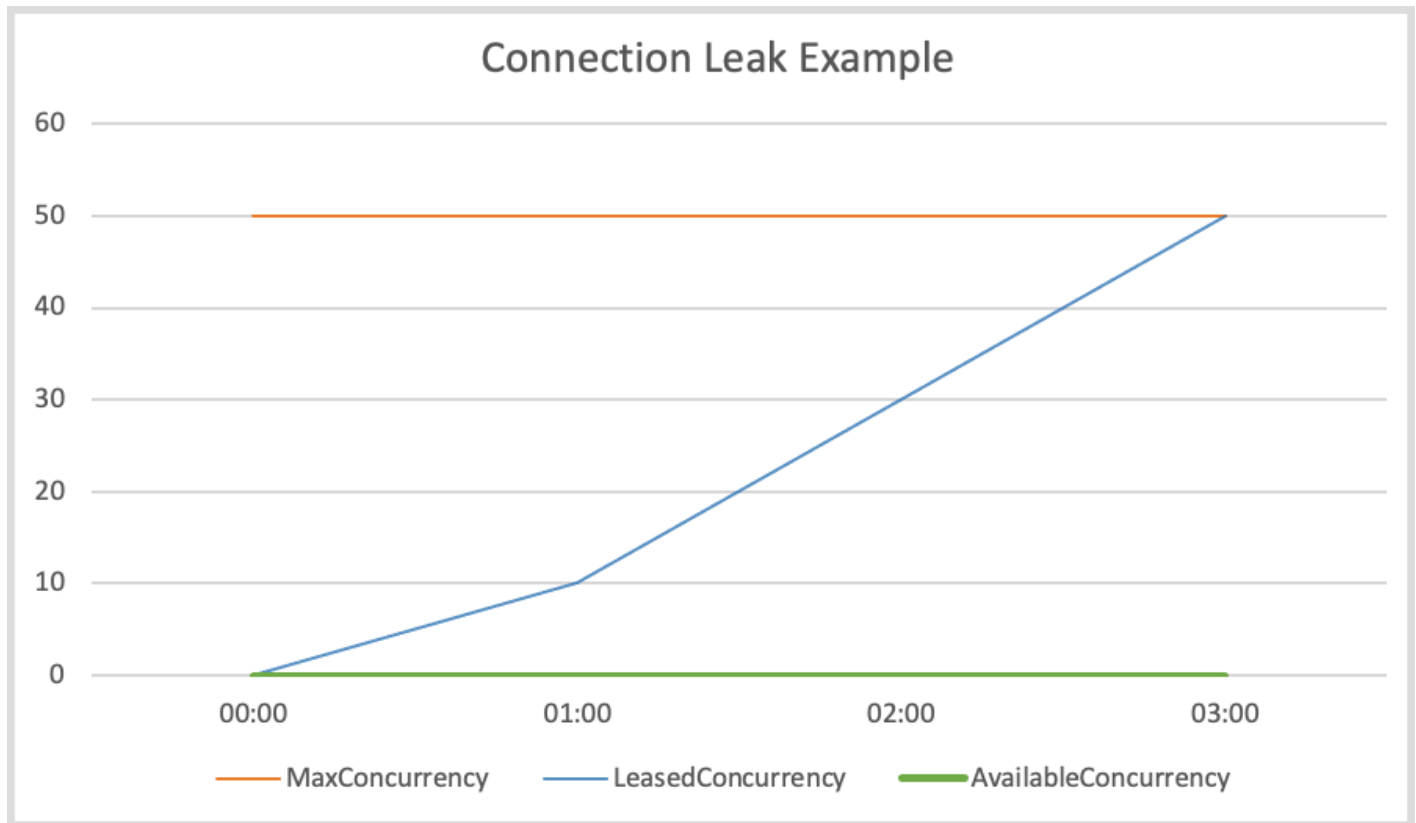
Intente ejecutar un volcado de TCP para una conexión afectada en el extremo del cliente de la red (por ejemplo, después de cualquier servidor proxy que controle).

Si ves que el AWS punto final está enviando un mensaje TCP RST (restableciéndolo), [ponte en contacto con el servicio afectado](#). Prepárate para proporcionar los identificadores de solicitud y las marcas horarias que indiquen cuándo se produjo el problema. El equipo de AWS soporte también podría beneficiarse de [los registros de cableado](#) que muestren exactamente qué bytes envía y recibe su aplicación y cuándo.

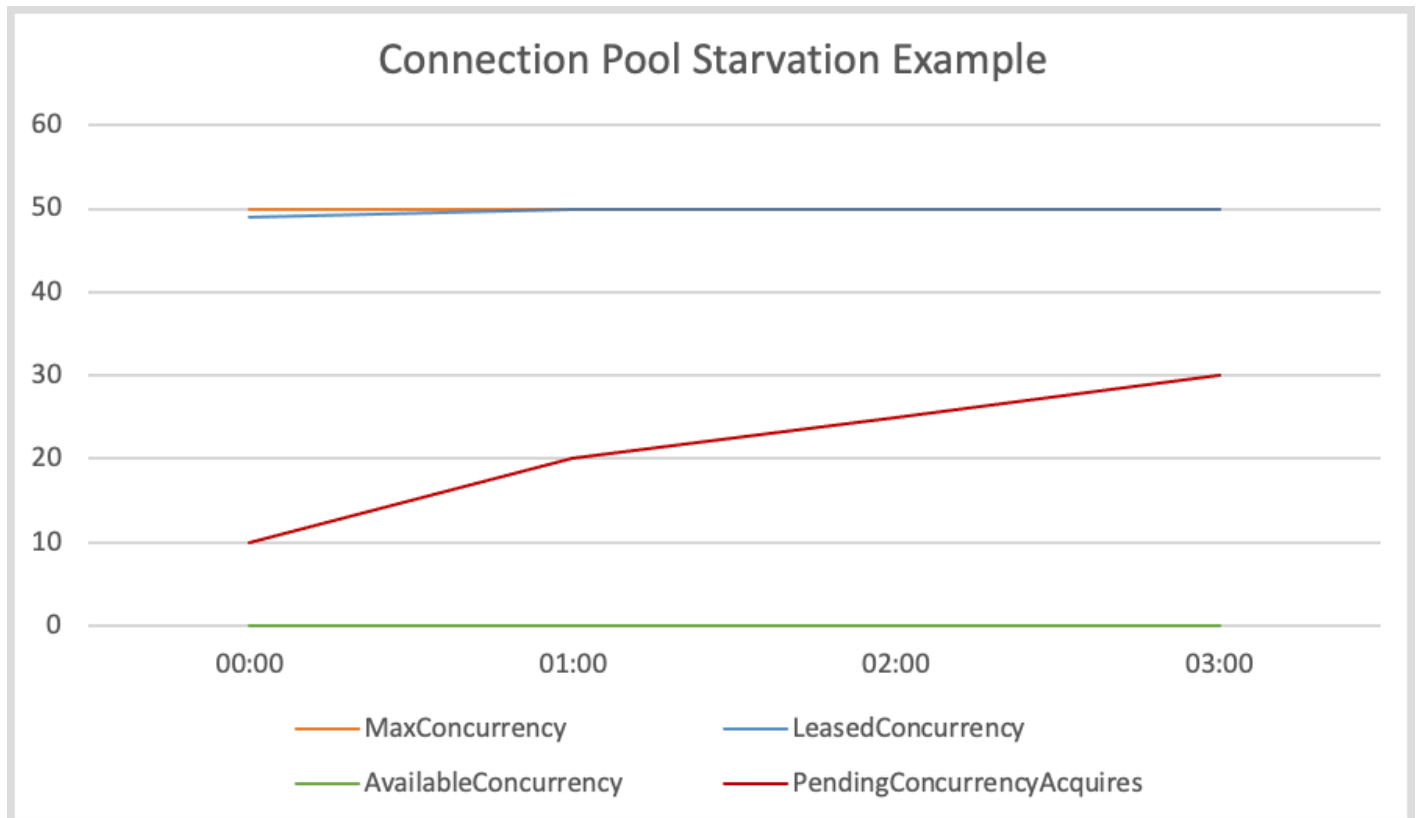
¿Cómo puedo solucionar el error «No se puede ejecutar la solicitud HTTP: se ha agotado el tiempo de espera para la conexión desde el grupo»?

Este error indica que una solicitud no puede obtener una conexión del grupo dentro del tiempo máximo especificado. Para solucionar el problema, te recomendamos que [habilites las métricas del lado del cliente del SDK para publicar las métricas](#) en Amazon. CloudWatch Las métricas HTTP pueden ayudar a reducir la causa raíz. En los siguientes elementos se describen las causas más comunes de este error.

- Pérdida de conexión. Puedes investigar esto comprobando `LeasedConcurrency`, `AvailableConcurrency`, y `MaxConcurrency` las métricas. Si `LeasedConcurrency` aumenta hasta alcanzar el límite `MaxConcurrency` pero nunca disminuye, es posible que se produzca una pérdida de conexión. Una causa común de una fuga es que una operación de streaming, como un `getObject` método S3, no está cerrada. Recomendamos que la aplicación lea todos los datos del flujo de entrada lo antes posible y [cierre el](#) flujo de entrada después. En el siguiente gráfico, se muestra el aspecto que podrían tener las métricas del SDK en caso de pérdida de conexión.



- Falta de agua en la piscina de conexiones. Esto puede suceder si la tasa de solicitudes es demasiado alta y el tamaño del grupo de conexiones que se ha configurado no puede satisfacer la demanda de la solicitud. El tamaño del grupo de conexiones predeterminado es 50 y, cuando las conexiones del grupo alcanzan el máximo, el cliente HTTP pone en cola las solicitudes entrantes hasta que las conexiones estén disponibles. En el siguiente gráfico, se muestra el aspecto que podrían tener las métricas del SDK en caso de escasez de grupos de conexiones.



Para mitigar este problema, considera la posibilidad de tomar alguna de las siguientes medidas.

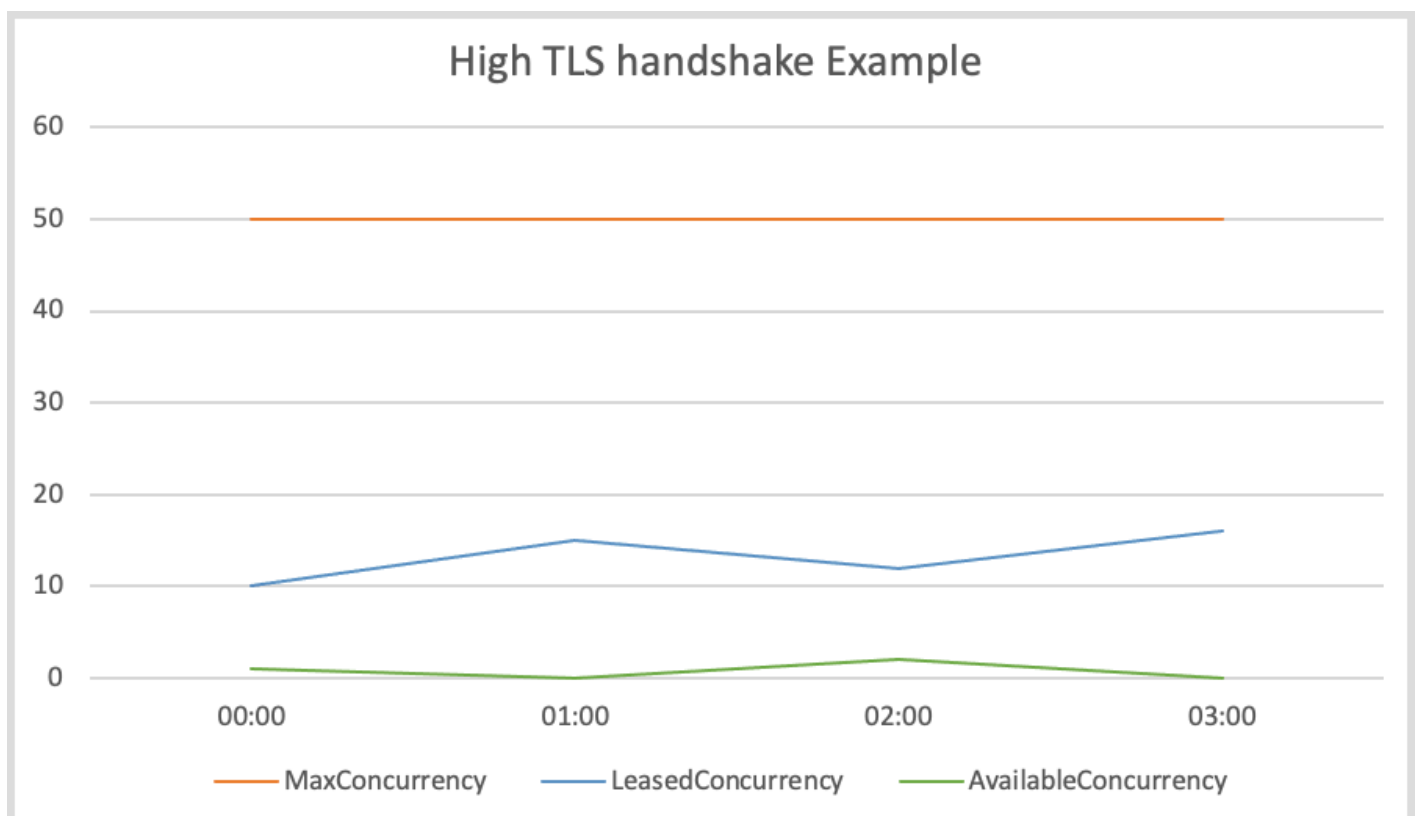
- Aumente el tamaño del grupo de conexiones,
- Aumente el tiempo de espera de adquisición.
- Disminuya la tasa de solicitudes.

Al aumentar el número máximo de conexiones, el rendimiento del cliente puede aumentar (a menos que la interfaz de red ya esté plenamente utilizada). Sin embargo, con el tiempo, es posible que el sistema operativo limite la cantidad de descriptores de archivos utilizados por el proceso. Si ya utiliza completamente la interfaz de red o no puede aumentar aún más el número de conexiones, intente aumentar el tiempo de espera de adquisición. Con este aumento, se gana tiempo adicional para las solicitudes de adquisición de una conexión antes de que se agote el tiempo de espera. Si las conexiones no se liberan, se agotará el tiempo de espera de las solicitudes subsiguientes.

Si no puedes solucionar el problema mediante los dos primeros mecanismos, reduce la tasa de solicitudes probando las siguientes opciones.

- Simplifique sus solicitudes para que las grandes ráfagas de tráfico no sobrecarguen al cliente.
- Sea más eficiente con las llamadas a Servicios de AWS.

- Aumente el número de anfitriones que envían solicitudes.
- Los subprocesos de E/S están demasiado ocupados. Esto solo se aplica si utiliza un cliente SDK asíncrono con. [NettyNioAsyncHttpClient](#) Si la `AvailableConcurrency` métrica no es baja (lo que indica que hay conexiones disponibles en el grupo), pero `ConcurrencyAcquireDuration` es alta, podría deberse a que los subprocesos de E/S no pueden gestionar las solicitudes. Asegúrese de no hacerse pasar por `Runnable:run` un [ejecutor de finalización futura](#) y de realizar una tarea que requiera mucho tiempo en la cadena de finalización futura de la respuesta, ya que esto puede bloquear un subproceso de E/S. Si ese no es el caso, considere la posibilidad de aumentar el número de subprocesos de E/S mediante el método. [eventLoopGroupBuilder](#) Como referencia, el número predeterminado de subprocesos de E/S para una `NettyNioAsyncHttpClient` instancia es el doble del número de núcleos de CPU del host.
- Alta latencia del protocolo de enlace TLS. Si tu `AvailableConcurrency` métrica está cerca de 0 y `LeasedConcurrency` es inferior a `MaxConcurrency`, puede deberse a que la latencia del protocolo de enlace TLS es alta. En el siguiente gráfico, se muestra el aspecto que podrían tener las métricas del SDK en caso de una latencia alta del protocolo de enlace de TLS.



En el caso de los clientes HTTP que ofrece el SDK de Java que no están basados en CRT, intenta habilitar los [registros de TLS para solucionar problemas de TLS](#). [Para el cliente HTTP AWS](#)

[basado en CRT, intente habilitar los registros CRT.AWS Si observa que el AWS punto final tarda mucho en realizar un protocolo de enlace TLS, póngase en contacto con el servicio afectado.](#)

¿Cómo puedo solucionar un `NoClassDefFoundError`, `NoSuchMethodError` o `NoSuchFieldError`?

A `NoClassDefFoundError` indica que no se pudo cargar una clase en tiempo de ejecución. Las dos causas más comunes de este error son:

- la clase no existe en la ruta de clases porque falta el JAR o porque hay una versión incorrecta del JAR en la ruta de clases.
- la clase no se pudo cargar porque su inicializador estático generó una excepción.

Del mismo modo, `NoSuchMethodError` s y `NoSuchFieldError` s suelen ser el resultado de una versión JAR que no coincide. Le recomendamos que lleve a cabo los siguientes pasos.

1. Comprueba tus dependencias para asegurarte de que estás usando la misma versión de todos los tarros del SDK. La razón más común por la que no se encuentra una clase, un método o un campo es cuando actualizas a una nueva versión de cliente pero sigues usando una versión antigua de dependencia del SDK «compartida». Es posible que la nueva versión del cliente intente usar clases que solo existen en las dependencias del SDK «compartidas» más recientes. Intenta ejecutar `mvn dependency:tree` o `gradle dependencies` (en el caso de Gradle) verifica que todas las versiones de la biblioteca del SDK coincidan. Para evitar este problema por completo en el futuro, recomendamos usar [BOM \(lista de materiales\) para administrar las versiones de](#) los módulos del SDK.

El siguiente ejemplo muestra un ejemplo de versiones mixtas del SDK.

```
[INFO] +- software.amazon.awssdk:dynamodb:jar:2.20.00:compile
[INFO] | +- software.amazon.awssdk:aws-core:jar:2.13.19:compile
[INFO] +- software.amazon.awssdk:netty-nio-client:jar:2.20.00:compile
```

La versión de `dynamodb` es la 2.20.00 y la versión de `aws-core` es la 2.13.19. La versión del `aws-core` artefacto también debería ser la 2.20.00.

2. Comprueba las sentencias al principio de tus registros para ver si una clase no se carga debido a un error de inicialización estática. La primera vez que la clase no se carga, puede generar

una excepción diferente y más útil que especifique por qué no se puede cargar la clase. Esta excepción, potencialmente útil, se produce solo una vez, por lo que las declaraciones de registro posteriores solo informarán de que no se ha encontrado la clase.

3. Compruebe el proceso de despliegue para asegurarse de que realmente implementa los archivos JAR necesarios junto con la aplicación. Es posible que esté compilando con la versión correcta, pero el proceso que crea la ruta de clases para su aplicación excluye una dependencia obligatoria.

¿Cómo puedo corregir un error «**SignatureDoesNotMatch**» o un error «La firma de la solicitud que calculamos no coincide con la firma que has proporcionado»?

Un `SignatureDoesNotMatch` error indica que la firma generada por el AWS SDK for Java y la firma generada por el Servicio de AWS no coinciden. Los siguientes elementos describen las posibles causas.

- Un apoderado o un intermediario modifica la solicitud. Por ejemplo, un proxy o un balanceador de cargas puede modificar un encabezado, una ruta o una cadena de consulta que haya sido firmada por el SDK.
- El servicio y el SDK se diferencian en la forma en que codifican la solicitud cuando cada uno genera la cadena que se va a firmar.

Para solucionar este problema, te recomendamos que [habilites el registro de depuración](#) en el SDK. Intenta reproducir el error y busca la solicitud canónica que generó el SDK. En el registro, la solicitud canónica se etiqueta con `AWS4 Canonical Request: ...` y la cadena a firmar se etiqueta con `AWS4 String to sign: ...`

Si no puedes habilitar la depuración (por ejemplo, porque solo se puede reproducir en producción), agrega una lógica a tu aplicación que registre la información sobre la solicitud cuando se produzca el error. A continuación, puedes usar esa información para intentar replicar el error fuera del entorno de producción en una prueba de integración con el registro de depuración activado.

Una vez recopiladas la solicitud canónica y la cadena de firma, compárelas con la [especificación de la versión 4 de AWS Signature](#) para determinar si hay algún problema en la forma en que el SDK generó la cadena para firmarla. Si algo parece estar mal, puedes crear un [informe de GitHub error](#) para AWS SDK for Java

Si no hay ningún problema, puedes comparar la cadena del SDK para firmar con la cadena para firmar que algunos Servicios de AWS devuelven como parte de la respuesta al error (Amazon S3, por ejemplo). Si no está disponible, ponte en [contacto con el servicio afectado](#) para ver qué solicitud canónica y cadena de firma generaron para poder comparar. Estas comparaciones pueden ayudar a identificar a las partes intermediarias que podrían haber modificado la solicitud o a codificar las diferencias entre el servicio y el cliente.

Para obtener más información general sobre la firma de solicitudes, consulte [Firmar solicitudes de AWS API](#) en la Guía del AWS Identity and Access Management usuario.

Example de una solicitud canónica

```
PUT
/Example-Bucket/Example-Object
partNumber=19&uploadId=string
amz-sdk-invocation-id:f8c2799d-367c-f024-e8fa-6ad6d0a1afb9
amz-sdk-request:attempt=1; max=4
content-encoding:aws-chunked
content-length:51
content-type:application/octet-stream
host:xxxxx
x-amz-content-sha256:STREAMING-UNSIGNED-PAYLOAD-TRAILER
x-amz-date:20240308T034733Z
x-amz-decoded-content-length:10
x-amz-sdk-checksum-algorithm:CRC32
x-amz-trailer:x-amz-checksum-crc32
```

Example de una cadena para firmar

```
AWS4-HMAC-SHA256
20240308T034435Z
20240308/us-east-1/s3/aws4_request
5f20a7604b1ef65dd89c333fd66736fdef9578d11a4f5d22d289597c387dc713
```

¿Cómo puedo solucionar el error

«**java.lang.IllegalStateException**: El grupo de conexiones se ha cerrado»?

Este error indica que el grupo de conexiones HTTP de Apache subyacente estaba cerrado. Los siguientes elementos describen las posibles causas.

- El cliente del SDK se cerró prematuramente. El SDK solo cierra el grupo de conexiones cuando el cliente asociado está cerrado. Asegúrese de no cerrar los recursos mientras estén en uso.
- Se **java.lang.Error** arrojó un. Errores como el que `OutOfMemoryError` provocan el [cierre](#) de un grupo de conexiones HTTP de Apache. Examine sus registros para ver si hay rastros de errores en las pilas. Revisa también el código para ver los lugares en los que capta `Throwable`s o `Errors`, pero se traga el resultado que evita que aparezca el error. Si tu código no muestra errores, vuelve a escribirlo para que se registre la información. La información registrada ayuda a determinar la causa raíz del error.
- Intentó utilizar el proveedor de credenciales del que se devolvió **`DefaultCredentialsProvider#create()`** después de cerrarlo. [`DefaultCredentialsProvider#create\(\)`](#) devuelve una instancia única, por lo que si está cerrada y tu código llama al `resolveCredentials` método, la excepción se generará una vez que caduquen las credenciales (o el token) en caché.

Comprueba el código para ver los lugares donde `DefaultCredentialsProvider` está cerrado, como se muestra en los siguientes ejemplos.

- La instancia de Singleton se cierra llamando `DefaultCredentialsProvider#close()`.

```
DefaultCredentialsProvider defaultCredentialsProvider =
    DefaultCredentialsProvider.create(); // Singleton instance returned.
AwsCredentials credentials = defaultCredentialsProvider.resolveCredentials();

// Make calls to Servicios de AWS.

defaultCredentialsProvider.close(); // Explicit close.

// Make calls to Servicios de AWS.

// After the credentials expire, either of the following calls eventually results
// in a "Connection pool shut down" exception.
credentials = defaultCredentialsProvider.resolveCredentials();
// Or
credentials = DefaultCredentialsProvider.create().resolveCredentials();
```

- Invoca `DefaultCredentialsProvider#create()` en un try-with-resources bloque.

```
try (DefaultCredentialsProvider defaultCredentialsProvider =
    DefaultCredentialsProvider.create()) {
    AwsCredentials credentials = defaultCredentialsProvider.resolveCredentials();
```



```
// Make calls to Servicios de AWS.

} // After the try-with-resources block exits, the singleton
  DefaultCredentialsProvider is closed.

// Make calls to Servicios de AWS.

DefaultCredentialsProvider defaultCredentialsProvider =
  DefaultCredentialsProvider.create(); // The closed singleton instance is returned.
// If the credentials (or token) has expired, the following call results in the
  error.
AwsCredentials credentials = defaultCredentialsProvider.resolveCredentials();
```

Cree una nueva instancia que no sea singleton llamando `DefaultCredentialsProvider.builder().build()` si su código ha cerrado la instancia singleton y necesita resolver las credenciales mediante un `DefaultCredentialsProvider`

Utilice las funciones de la versión AWS SDK for Java 2.x

Características generales

El SDK para Java 2.x contiene varias funciones que facilitan la programación con Servicios de AWS .

- El SDK oculta los complejos mecanismos que hay detrás de la [recuperación de los resultados paginados](#) y el [sondeo de recursos](#).
- La [programación asíncrona con E/S sin bloqueo](#) le ayuda a escribir código simultáneo con un mejor rendimiento. El SDK ofrece las ventajas de [HTTP/2](#), como la reducción de la latencia, siempre que sea posible.
- El SDK de Java puede generar [métricas](#) para ayudarle a supervisar el estado operativo de sus aplicaciones.

Características específicas de servicios

Además de las funciones generales mencionadas anteriormente, el SDK de Java proporciona funciones específicas. Servicios de AWS

- Amazon S3: para [simplificar el trabajo con archivos y directorios](#) con Amazon S3, el SDK incluye el S3 Transfer Manager. Para [mejorar el rendimiento y la fiabilidad](#) al utilizar la API S3 asíncrona estándar del SDK, el SDK ofrece el cliente S3 basado en AWS CRT.
- DynamoDB: la API de cliente mejorado de DynamoDB proporciona la [capacidad de asignación orientada a objetos](#). [Trabaje con datos de estilo JSON y orientados a documentos](#) mediante la API de documentos mejorada.
- IAM: la API de IAM Policy Builder proporciona una [forma segura de escribir y orientada a objetos para crear políticas de IAM](#).

Trabaje con resultados paginados mediante la versión 2.x de AWS SDK for Java

Muchas AWS operaciones devuelven resultados paginados cuando el objeto de respuesta es demasiado grande para mostrarlos en una sola respuesta. En la AWS SDK for Java versión 1.0, la respuesta contiene un token que se usa para recuperar la siguiente página de resultados. Por el

contrario, la versión AWS SDK for Java 2.x cuenta con métodos de autopaginación que permiten realizar múltiples llamadas de servicio para obtener automáticamente la siguiente página de resultados. Solo tiene que escribir el código que procesa los resultados. La paginación automática está disponible para clientes síncronos y asíncronos.

Note

Estos fragmentos de código asumen que entiende [los fundamentos del uso del SDK](#), y ha configurado su entorno con [acceso de inicio de sesión único](#).

Paginación síncrona

En los ejemplos siguientes, se muestran métodos de paginación síncrona para obtener una lista de los objetos en un bucket de Amazon S3 .

Iterar sobre páginas

El primer ejemplo muestra el uso de un objeto `ListRes` paginador, una [ListObjectsV2Iterable](#) instancia, para recorrer en iteración todas las páginas de respuesta con el método `stream`. El código recorre las páginas de respuesta, convierte el flujo de respuesta en un flujo de [S3Object](#) contenido y, a continuación, procesa el contenido del objeto. Amazon S3

Las importaciones siguientes se aplican a todos los ejemplos de esta sección de paginación síncrona.

Importaciones

```
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()));
```

Consulte el [ejemplo completo](#) en GitHub.

Iterar sobre objetos

Los ejemplos siguientes muestran formas de recorrer en iteración los objetos devueltos en la respuesta en lugar de las páginas de la respuesta. El método `contents` de clase `ListObjectsV2Iterable` devuelve un [SdkIterable](#) que proporciona varios métodos para procesar los elementos de contenido subyacentes.

Usar un stream

El siguiente fragmento de código utiliza el método `stream` sobre el contenido de la respuesta para recorrer en iteración la colección de elementos 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()));
```

Consulte el [ejemplo completo](#) en GitHub.

Usar un bucle for-each

Como `SdkIterable` amplía la interfaz de `Iterable`, puede procesar el contenido como cualquier otro `Iterable`. El siguiente fragmento de código utiliza un bucle estándar `for-each` para recorrer en iteración el contenido de la respuesta.

```
for (S3Object content : listRes.contents()) {
    System.out.println(" Key: " + content.key() + " size = " +
content.size());
}
```

Consulte el [ejemplo completo](#) en GitHub.

Paginación manual

Si su caso de uso lo requiere, la paginación manual seguirá estando disponible. Utilice el siguiente token del objeto de respuesta para las solicitudes posteriores. En este ejemplo se usa un bucle `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();
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Paginación asíncrona

Los siguientes ejemplos muestran los métodos de paginación asíncrona para enumerar tablas. DynamoDB

Recorrer en iteración páginas de nombres de tablas

Los dos ejemplos siguientes utilizan un cliente DynamoDB asíncrono que llama `listTablesPaginator` al método con una solicitud para obtener un [ListTablesPublisher](#). `ListTablesPublisher` implementa dos interfaces, lo que proporciona muchas opciones para procesar las respuestas. Analizaremos los métodos de cada interfaz.

Utilizar un **Subscriber**.

En el ejemplo de código siguiente se muestra cómo procesar los resultados paginados mediante la interfaz `org.reactivestreams.Publisher` implementada por `ListTablesPublisher`. Para obtener más información sobre el modelo de flujos reactivos, consulte el [GitHub repositorio de flujos reactivos](#).

Las importaciones siguientes se aplican a todos los ejemplos de esta sección de paginación asíncrona.

Importaciones

```
import io.reactivex.rxjava3.core.Flowable;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import reactor.core.publisher.Flux;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
```

```
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.paginators.ListTablesPublisher;

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

El siguiente código adquiere una instancia `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);
```

El código siguiente utiliza una implementación anónima de `org.reactivestreams.Subscriber` para procesar los resultados de cada página.

El método `onSubscribe` llama al método `Subscription.request` para iniciar las solicitudes de datos del publicador. Este método debe llamarse para empezar a obtener datos del publicador.

El método `onNext` del suscriptor procesa una página de respuesta accediendo a todos los nombres de las tablas e imprimiendo cada una de ellas. Una vez procesada la página, se solicita otra página al publicador. Este es el método que se llama repetidamente hasta que se recuperan todas las páginas.

El método `onError` se desencadena si se produce un error al recuperar los datos. Por último, se llama al método `onComplete` cuando se han solicitado todas las páginas.

```
// 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.
        }
    });
```

Consulta el [ejemplo completo](#) en GitHub

Utilizar un **Consumer**.

La interfaz `SdkPublisher` que `ListTablesPublisher` implementa tiene un método `subscribe` que toma un `Consumer` y devuelve un `CompletableFuture<Void>`.

El método `subscribe` de esta interfaz se puede utilizar para casos de uso simples en los que un `org.reactivestreams.Subscriber` puede resultar demasiado recargado. Como el siguiente código consume cada página, llama al método `tableNames` en cada una de ellas. El método `tableNames` devuelve un `java.util.List` de nombres de tablas de DynamoDB que se procesan con el método `forEach`.


```
// Use a Consumer for simple use cases.
CompletableFuture<Void> future = publisher.subscribe(
    response -> response.tableNames()
        .forEach(System.out::println));
```

Consulte el [ejemplo completo](#) en GitHub.

Recorrer en iteración nombres de tablas

Los ejemplos siguientes muestran formas de recorrer en iteración los objetos devueltos en la respuesta en lugar de las páginas de la respuesta. De forma similar al ejemplo síncrono de Amazon S3 mostrado anteriormente con su método `contents`, la clase de resultados asíncrona de DynamoDB, `ListTablesPublisher` dispone del método de conveniencia `tableNames` para interactuar con la colección de elementos subyacente. El tipo devuelto del método `tableNames` es un [SdkPublisher](#) que se puede usar para solicitar elementos en todas las páginas.

Utilizar un **Subscriber**.

El código siguiente adquiere un `SdkPublisher` de las colecciones subyacentes de nombres de tablas.

```
// 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();
```

El código siguiente utiliza una implementación anónima de `org.reactivestreams.Subscriber` para procesar los resultados de cada página.

El método del suscriptor `onNext` procesa un elemento individual de la colección. En este caso, es el nombre de una tabla. Una vez procesada la página, se solicita otra página al publicador. Este es el método que se llama repetidamente hasta que se recuperan todas las páginas.

```
// 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() {
}
});
```

Consulte el [ejemplo completo](#) en GitHub.

Utilizar un **Consumer**.

En el ejemplo siguiente, se utiliza el método `subscribe` de `SdkPublisher` que toma un `Consumer` a para procesar cada elemento.

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

Consulte el [ejemplo completo](#) en GitHub.

Usar una biblioteca de terceros

Puede utilizar otras bibliotecas de terceros en lugar de implementar un suscriptor personalizado. Este ejemplo demuestra el uso de RxJava, pero se puede utilizar cualquier biblioteca que implemente las interfaces de flujo reactivo. Consulte la [página RxJava wiki GitHub](#) para obtener más información sobre esa biblioteca.

Para utilizar la biblioteca, añádala como una dependencia. Si utiliza Maven, el ejemplo muestra el fragmento de POM que se debe utilizar.

Entrada de 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);
```

Consulte el [ejemplo completo](#) en GitHub.

Encuesta sobre los estados de los recursos en la versión AWS SDK for Java 2.x: Waiters

La utilidad waiters de la versión AWS SDK for Java 2.x permite validar que AWS los recursos se encuentran en un estado específico antes de realizar operaciones con esos recursos.

Un camarero es una abstracción que se utiliza para sondear AWS recursos, como DynamoDB tablas o Amazon S3 cubos, hasta que se alcance el estado deseado (o hasta que se determine que el recurso nunca alcanzará el estado deseado). En lugar de escribir una lógica para sondear continuamente AWS los recursos, lo que puede resultar engorroso y propenso a errores, puede utilizar waiters para sondear un recurso y hacer que el código siga ejecutándose una vez que el recurso esté listo.

Requisitos previos

[Para poder usar camareros en un proyecto con el AWS SDK for Java, debes completar los pasos que se indican en Configuración de la versión 2.x. AWS SDK for Java](#)

También debe configurar las dependencias de su proyecto (por ejemplo, en su archivo `pom.xml` o `build.gradle`) para usar la versión 2.15.0 o posterior del AWS SDK for Java.

Por ejemplo:

```
<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>
```

Uso de esperadores

Para crear una instancia de un objeto esperador, cree antes un cliente de servicio. Establece el método del cliente `waiter()` de servicio como el valor del objeto esperador. Una vez que exista la instancia de esperador, configure sus opciones de respuesta para ejecutar el código apropiado.

Programación asíncrona

En el siguiente fragmento de código se muestra cómo esperar a que una DynamoDB mesa exista y se encuentre en estado ACTIVO.

```
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);
```

Programación asíncrona

El siguiente fragmento de código muestra cómo esperar a que una DynamoDB tabla deje de existir.

```
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 los esperadores

Puede personalizar la configuración de un esperador utilizando el `overrideConfiguration()` en su creador. Para algunas operaciones, puede aplicar una configuración personalizada al realizar la solicitud.

Configurar un esperador

En el siguiente fragmento de código se muestra cómo anular la configuración de un esperador.

```
// 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();
```

Anular la configuración de una solicitud específica

El siguiente fragmento de código muestra cómo anular la configuración de un esperador en función de cada solicitud. Tenga en cuenta que solo algunas operaciones tienen configuraciones personalizables.

```
waiter.waitUntilTableNotExists(b -> b.tableName("myTable"),  
    o -> o.maxAttempts(10));  
  
asyncWaiter.waitUntilTableExists(b -> b.tableName("myTable"),  
    o -> o.waitTimeout(Duration.ofMinutes(1)));
```

Ejemplos de código

Para ver un ejemplo completo del uso de waiters with DynamoDB, consulta [CreateTable.java](#) en el AWS repositorio de ejemplos de código.

Para ver un ejemplo completo del uso de waiters with Amazon S3, consulta [S3 BucketOps .java](#) en el AWS repositorio de ejemplos de código.

Usar la programación asíncrona

AWS SDK for Java 2.x Cuenta con clientes asíncronos con soporte de E/S sin bloqueo que implementan una alta concurrencia en unos pocos subprocesos. Sin embargo, no se garantiza una E/S total sin bloqueo. Un cliente asíncrono puede bloquear las llamadas en algunos casos, como la recuperación de credenciales, la firma de solicitudes mediante la [versión 4 de AWS Signature \(Sigv4\)](#) o la detección de terminales.

Los métodos síncronos bloquean la ejecución de los subprocesos hasta que el cliente recibe una respuesta del servicio. Los métodos asíncronos terminan de ejecutarse inmediatamente, devolviendo el control al subproceso que realiza la llamada sin esperar una respuesta.

Como un método asíncrono termina de ejecutarse antes de que haya una respuesta disponible, necesita una forma de obtener la respuesta cuando esté lista. Los métodos para el cliente asíncrono

de la versión 2.x de los `CompletableFuture` objetos AWS SDK for Java devuelven objetos que permiten acceder a la respuesta cuando esté lista.

Operaciones sin streaming

Para las operaciones sin streaming, las llamadas a métodos asíncronos son similares a los métodos síncronos. Sin embargo, los métodos asíncronos AWS SDK for Java devuelven un [CompletableFuture](#) objeto que contiene los resultados de la operación asíncronica en el futuro.

Llame al método `CompletableFuture whenComplete()` con una acción para completar cuando el resultado esté disponible. `CompletableFuture` implementa la interfaz de `Future`, por lo que también puede obtener el objeto de respuesta llamando al método `get()`.

El siguiente es un ejemplo de una operación asíncrona que llama a una Amazon DynamoDB función para obtener una lista de tablas y recibe una que puede contener un objeto. `CompletableFuture ListTablesResponse` La acción definida en la llamada a `whenComplete()` solo se realiza cuando se completa la llamada asíncrona.

Importaciones

```
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();
}
}
```

En el ejemplo de código siguiente se muestra cómo recuperar un elemento de una tabla mediante el cliente asíncrono. Invoque el `getItem` método de `DynamoDbAsyncClient` y pásele un [GetItemRequest](#) objeto con el nombre de la tabla y el valor de la clave principal del elemento que desee. Normalmente, esta es la forma en que se pasan los datos que requiere la operación. En este ejemplo, observe que se pasa un valor `String`.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub

Operaciones de streaming

Para las operaciones de streaming, debes proporcionar una [AsyncRequestBody](#) para proporcionar el contenido de forma incremental o una [AsyncResponseTransformer](#) para recibir y procesar la respuesta.

En el siguiente ejemplo, se carga un archivo de forma Amazon S3 asíncrona mediante la operación. `PutObject`

Importaciones

```
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();
}
```

En el siguiente ejemplo, se obtiene un archivo de forma Amazon S3 asíncrona mediante la operación. `GetObject`

Importaciones

```
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();
}
```

Operaciones avanzadas

La versión AWS SDK for Java 2.x utiliza [Netty](#), un marco de aplicaciones de red asíncrono controlado por eventos, para gestionar los subprocesos de E/S. El AWS SDK for Java 2.x crea un `ExecutorService` después de Netty para completar los futuros devueltos desde la solicitud del cliente HTTP al cliente Netty. Esta abstracción reduce el riesgo de que una aplicación interrumpa el proceso de sincronización si los desarrolladores deciden detener o suspender los subprocesos. De

forma predeterminada, cada cliente asíncrono crea un grupo de subprocesos en función del número de procesadores y gestiona las tareas de una cola dentro del `ExecutorService`.

Los usuarios avanzados pueden especificar el tamaño de grupo de subprocesos al crear un cliente asíncrono mediante la siguiente opción en la compilación.

Código

```
S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            Executors.newFixedThreadPool(10)
        )
    )
    .build();
```

Para optimizar el rendimiento, puede administrar su propio ejecutor de grupo de subprocesos e incluirlo al configurar el 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();
```

Trabaje con HTTP/2 en AWS SDK for Java

HTTP/2 es una revisión importante del protocolo HTTP. Esta nueva versión tiene varias mejoras para optimizar el rendimiento:

- La codificación de datos binarios permite una transferencia de datos más eficiente.
- La compresión de encabezado reduce la sobrecarga de bytes descargados por el cliente, lo que ayuda a que el cliente obtenga el contenido antes. Esto resulta especialmente útil para los clientes móviles que ya tienen un ancho de banda limitado.
- La comunicación asíncrona bidireccional (multiplexación) permite enviar múltiples solicitudes y mensajes de respuesta entre el cliente y transmitirlos al mismo tiempo AWS a través de una sola conexión, en lugar de hacerlo a través de múltiples conexiones, lo que mejora el rendimiento.

Los desarrolladores que actualizan a los SDK más recientes utilizarán automáticamente HTTP/2 cuando sea compatible con el servicio con el que estén trabajando. Las nuevas interfaces de programación aprovechan fácilmente las características de HTTP/2 y proporcionan nuevas formas de crear aplicaciones.

La versión AWS SDK for Java 2.x incluye nuevas API para la transmisión de eventos que implementan el protocolo HTTP/2. Para ver ejemplos de cómo usar estas nuevas API, consulte [Trabajar con Kinesis](#).

Utilice las métricas del SDK del AWS SDK for Java

Con la AWS SDK for Java versión 2.x, puede recopilar métricas sobre los clientes de servicio de su aplicación, analizar los resultados y Amazon CloudWatch, después, actuar en consecuencia.

De forma predeterminada, la recopilación de métricas está deshabilitada en el SDK. En este tema se explica cómo activarlo y configurarlo.

Requisitos previos

Antes de poder activar y utilizar las métricas, debe completar los siguientes pasos:

- Realice los pasos que se indican en [Configuración](#).
- Configure las dependencias de su proyecto (por ejemplo, en su archivo `pom.xml` o `build.gradle`) para usar la versión `2.14.0` o posterior del AWS SDK for Java.

Para permitir la publicación de métricas en CloudWatch, incluye también el `cloudwatch-metric-publisher` ArtifactID con el `2.14.0` número de versión o posterior en las dependencias de tu proyecto.

Por ejemplo:

```
<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 `cloudwatch:PutMetricData` los permisos para la identidad de IAM utilizada por el editor de métricas para permitir que el SDK para Java escriba métricas.

Habilitar la recopilación de métricas

Puede habilitar las métricas en su aplicación para un cliente de servicio o para solicitudes individuales.

Habilitar métricas para una solicitud específica

La siguiente clase muestra cómo habilitar el publicador de CloudWatch métricas para una solicitud. Amazon DynamoDB Utiliza la configuración predeterminada del publicador de métricas.

```
import software.amazon.awssdk.metrics.MetricPublisher;
```



```
import software.amazon.awssdk.metrics.publishers.cloudwatch.CloudWatchMetricPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;

public class DefaultConfigForRequest {
    // Use one MetricPublisher for your application. It can be used with requests or
    // service clients.
    static MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

    public static void main(String[] args) {
        DynamoDbClient ddb = DynamoDbClient.create();
        // Publish metrics the for ListTables operation.
        ddb.listTables(ListTablesRequest.builder()
            .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
            .build());

        // Perform more work in your application.

        // A MetricsPublisher has its own lifecycle independent of any service client
        // or request that uses it.
        // If you no longer need the publisher, close it to free up resources.
        metricsPub.close(); // All metrics stored in memory are flushed to CloudWatch.

        // Perform more work with the DynamoDbClient instance without publishing
        // metrics.
        // Close the service client when you no longer need it.
        ddb.close();
    }
}
```

Important

Asegúrese de que la aplicación llame `close` a la [MetricPublisher](#) instancia cuando el cliente del servicio ya no esté en uso. Si no lo hace, es posible que se filtre el subprocesso o el descriptor del archivo.

Habilite las métricas resumidas para un cliente de servicio específico

En el siguiente fragmento de código, se muestra cómo habilitar un publicador de CloudWatch métricas con la configuración predeterminada para un cliente de servicio.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

DynamoDbClient ddb = DynamoDbClient.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build();
```

Personalice el publicador de métricas

La siguiente clase muestra cómo configurar una configuración personalizada para el publicador de métricas de un cliente de servicio específico. Las personalizaciones incluyen cargar un perfil específico, especificar una AWS región a la que el publicador de métricas envía las solicitudes y personalizar la frecuencia a la que envía las métricas. CloudWatch

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.metrics.CoreMetric;
import software.amazon.awssdk.metrics.MetricPublisher;
import software.amazon.awssdk.metrics.publishers.cloudwatch.CloudWatchMetricPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import java.time.Duration;

public class CustomConfigForDDBClient {
    // Use one MetricPublisher for your application. It can be used with requests or
    // service clients.
    static MetricPublisher metricsPub = CloudWatchMetricPublisher.builder()
        .cloudWatchClient(CloudWatchAsyncClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create("cloudwatch"))
            .build())
        .uploadFrequency(Duration.ofMinutes(5))
        .maximumCallsPerUpload(100)
        .namespace("ExampleSDKV2Metrics")
        .detailedMetrics(CoreMetric.API_CALL_DURATION)
        .build();

    public static void main(String[] args) {
        DynamoDbClient ddb = DynamoDbClient.builder()
            .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
            .build();
        // Publish metrics for DynamoDB operations.
    }
}
```

```
    ddb.listTables();
    ddb.describeEndpoints();
    ddb.describeLimits();
    // Perform more work in your application.

    // A MetricsPublisher has its own lifecycle independent of any service client
    or request that uses it.
    // If you no longer need the publisher, close it to free up resources.
    metricsPub.close(); // All metrics stored in memory are flushed to CloudWatch.

    // Perform more work with the DynamoDbClient instance without publishing
    metrics.
    // Close the service client when you no longer need it.
    ddb.close();
}
}
```

Las personalizaciones que se muestran en el fragmento anterior tienen los siguientes efectos.

- El `cloudWatchClient` método permite personalizar el CloudWatch cliente que se utiliza para enviar las métricas. En este ejemplo, utilizamos una región diferente de la predeterminada `us-east-1` a la que el cliente envía las métricas. También utilizamos un perfil con nombre diferente, `cloudwatch`, cuyas credenciales se utilizarán para autenticar las solicitudes. CloudWatch Esas credenciales deben tener permisos para `cloudwatch:PutMetricData`
- El `uploadFrequency` método le permite especificar la frecuencia con la que el editor carga las métricas. CloudWatch El valor predeterminado es una vez por minuto.
- El `maximumCallsPerUpload` método limita el número de llamadas realizadas por carga. El valor predeterminado es ilimitado.
- De forma predeterminada, el SDK para Java 2.x publica las métricas en el espacio de nombres `AwsSdk/JavaSdk2`. Puede usar el `namespace` método para especificar un valor diferente.
- De forma predeterminada, el SDK publica métricas resumidas. Las métricas resumidas se componen del promedio, el mínimo, el máximo, la suma y el recuento de muestras. Al especificar una o más métricas del SDK en el `detailedMetrics` método, el SDK publica datos adicionales para cada métrica. Estos datos adicionales permiten realizar consultas con estadísticas de percentiles, como `p90` y `p99`. CloudWatch Las métricas detalladas son especialmente útiles para las métricas de `latenciaAPICallDuration`, como las que miden la end-to-end latencia de

las solicitudes de los clientes del SDK. Puedes usar los campos de la [CoreMetric](#) clase para especificar otras métricas comunes del SDK.

¿Cuándo están disponibles las métricas?

Por lo general, las métricas están disponibles entre 5 y 10 minutos después de que las emita el SDK para Java. Para obtener up-to-date métricas precisas, consulta Cloudwatch al menos 10 minutos después de emitir las métricas de tus aplicaciones Java.

¿Qué información se recopila?

La recopilación de métricas incluye lo siguiente:

- Número de solicitudes de API, se hayan realizado correctamente o no
- Información sobre los AWS servicios a los que llamas en tus solicitudes de API, incluidas las excepciones devueltas
- La duración de varias operaciones, como la serialización, la firma y las solicitudes HTTP
- Métricas del cliente HTTP, como el número de conexiones abiertas, el número de solicitudes pendientes y el nombre del cliente HTTP utilizado

Note

Las métricas disponibles varían según el cliente HTTP.

Para ver una lista completa, consulte [Métricas de los clientes de servicio](#).

¿Cómo puedo usar esta información?

Puede utilizar las métricas que recopila el SDK para monitorizar los clientes de servicio de su aplicación. Puede analizar las tendencias generales de uso, identificar anomalías, revisar las excepciones detectadas por los clientes del servicio o profundizar para entender un problema concreto. También puedes crear alarmas para que te notifiquen en cuanto tu aplicación alcance una condición que tú definas. Amazon CloudWatch

Para obtener más información, consulte [Uso de Amazon CloudWatch métricas](#) y [Uso de Amazon CloudWatch alarmas](#) en la [Guía del Amazon CloudWatch usuario](#).

Métricas de los clientes de servicio

Con él AWS SDK for Java 2.x, puedes recopilar métricas de los clientes de servicio de tu aplicación y, a continuación, publicarlas (generar) esas métricas en [Amazon CloudWatch](#).

En estas tablas, se muestran las métricas que puede recopilar y cualquier requisito de uso del cliente HTTP.

Para obtener más información sobre cómo habilitar y configurar las métricas del SDK, consulte [Habilitar métricas del SDK](#).

Métricas recopiladas con cada solicitud

Nombre de métrica	Descripción	Tipo
ApiCallDuration	El tiempo total que se tarda en finalizar una solicitud (incluidos todos los reintentos).	Duración
ApiCallSuccessful	Verdadero si la llamada a la API se ha realizado correctamente y falso si no.	Booleano
CredentialsFetchDuration	El tiempo que se tardó en obtener las credenciales de AWS firma de la solicitud.	Duración
EndpointResolveDuration	El tiempo que se tardó en resolver el punto final utilizado para la llamada a la API.	Duración
MarshallingDuration	El tiempo que se tarda en convertir una solicitud del SDK en una solicitud HTTP.	Duración
OperationName	El nombre de la AWS API a la que se realiza la solicitud.	Cadena

Nombre de métrica	Descripción	Tipo
RetryCount	Número de veces que el SDK volvió a intentar la llamada a la API.	Entero
ServiceId	ID de servicio con el Servicio de AWS que se realizó la solicitud de API.	Cadena
TokenFetchDuration	El tiempo que se tarda en obtener las credenciales de firma del token para la solicitud.	Duración

Métricas recopiladas para cada intento de solicitud

Es posible que cada llamada a la API requiera varios intentos antes de recibir una respuesta. Estas métricas se recogen para cada intento.

Métricas principales

Nombre de métrica	Descripción	Tipo
AwsExtendedRequestId	El identificador de solicitud ampliado de la solicitud de servicio.	Cadena
AwsRequestId	El identificador de solicitud de la solicitud de servicio.	Cadena
BackoffDelayDuration	El tiempo que el SDK esperó antes de este intento de llamada a la API.	Duración
ErrorType	El tipo de error que se produjo al intentar una llamada.	Cadena

Nombre de métrica	Descripción	Tipo
ReadThroughput	El rendimiento de lectura del cliente.	Doble
ServiceCallDuration	El tiempo que se tarda en conectarse al servicio, enviar la solicitud y recibir el código de estado HTTP y el encabezado de la respuesta.	Duración
SigningDuration	El tiempo que se tarda en firmar la solicitud HTTP.	Duración
TimeToFirstByte	Tiempo transcurrido desde el envío de la solicitud HTTP (incluida la adquisición de una conexión) hasta la recepción del primer byte de los encabezados de la respuesta.	Duración
TimeToLastByte	Tiempo transcurrido desde el envío de la solicitud HTTP (incluida la adquisición de una conexión) hasta la recepción del último byte de la respuesta .	Duración
UnmarshallingDuration	El tiempo que se tarda en convertir una respuesta HTTP en una respuesta de SDK.	Duración

Métricas HTTP

Nombre de métrica	Descripción	Tipo	Se requiere un cliente HTTP*
AvailableConcurrency	El número de solicitudes simultáneas restantes que puede admitir el cliente HTTP sin necesidad de establecer otra conexión.	Entero	Apache, Netty, CRT
ConcurrencyAcquireDuration	El tiempo que se tarda en adquirir un canal del grupo de conexiones.	Duración	Apache, Netty, CRT
HttpClientName	El nombre del HTTP que se utiliza para la solicitud.	Cadena	Apache, Netty, CRT
HttpStatusCode	El código de estado devuelto con la respuesta HTTP.	Entero	Cualquiera
LeasedConcurrency	El número de solicitudes que el cliente HTTP está ejecutando actualmente.	Entero	Apache, Netty, CRT
LocalStreamWindowSize	El tamaño de la ventana HTTP/2 local en bytes del flujo en el que se ejecutó esta solicitud.	Entero	Netty

Nombre de métrica	Descripción	Tipo	Se requiere un cliente HTTP*
MaxConcurrency	El número máximo de solicitudes simultáneas que admite el cliente HTTP.	Entero	Apache, Netty, CRT
PendingConcurrency Acquires	El número de solicitudes bloqueadas que esperan a que haya otra conexión TCP o una nueva transmisión disponible en el grupo de conexiones.	Entero	Apache, Netty, CRT
RemoteStreamWindowSize	El tamaño de la ventana HTTP/2 remota en bytes del flujo en el que se ejecutó esta solicitud.	Entero	Netty

Los términos utilizados en la columna significan:

- Apache: el cliente HTTP basado en Apache ([ApacheHttpClient](#))
- Netty: el cliente HTTP basado en Netty ([NettyNioAsyncHttpClient](#))
- CRT: el cliente HTTP AWS basado en CRT ([AwsCrtAsyncHttpClient](#))
- Cualquiera: la recopilación de datos métricos no depende del cliente HTTP; esto incluye el cliente HTTP basado en URLConnection ([URLConnectionHttpClient](#))

Trabaje con el Servicios de AWS uso del AWS SDK for Java 2.x

En esta sección se proporcionan tutoriales breves y orientación sobre cómo trabajar con Select Servicios de AWS. Para ver un conjunto completo de ejemplos, consulta la [sección de ejemplos de código](#).

Temas

- [Trabajar con CloudWatch](#)
- [AWS servicios de bases de datos y AWS SDK for Java 2.x](#)
- [Trabaja con DynamoDB](#)
- [Trabaja con Amazon EC2](#)
- [Trabajar con IAM](#)
- [Trabaja con Kinesis](#)
- [Invocar, enumerar y eliminar funciones AWS Lambda](#)
- [Trabajo con Amazon S3](#)
- [Trabajar con Amazon Simple Notification Service](#)
- [Trabaja con Amazon Simple Queue Service](#)
- [Trabajar con Amazon Transcribe](#)

Trabajar con CloudWatch

En esta sección se proporcionan ejemplos de programación de [Amazon CloudWatch](#) mediante AWS SDK for Java 2.x.

Amazon CloudWatchmonitiza los recursos y las aplicaciones de Amazon Web Services (AWS) que ejecuta en AWS en tiempo real. Puede utilizar CloudWatch para recopilar y hacer un seguimiento de métricas, que son las variables que puede medir en sus recursos y aplicaciones. Las alarmas de CloudWatch envían notificaciones o realizan cambios automáticamente en los recursos que está supervisando en función de las reglas que defina.

Los siguientes ejemplos incluyen únicamente el código necesario para demostrar cada técnica. El [código de ejemplo completo está disponible en GitHub](#). Desde allí, puede descargar un único archivo

de código fuente o clonar el repositorio localmente para obtener todos los ejemplos para compilarlos y ejecutarlos.

Temas

- [Obtener métricas de CloudWatch](#)
- [Publicar datos de métricas personalizadas para CloudWatch](#)
- [Uso de alarmas de CloudWatch](#)
- [Usa Amazon CloudWatch Events](#)

Obtener métricas de CloudWatch

Mostrar métricas

Para enumerar CloudWatch las métricas, crea un `listMetrics` método [ListMetricsRequest](#) `CloudWatchClient` llama al. Puede utilizar el objeto `ListMetricsRequest` para filtrar las métricas devueltas por espacio de nombres, nombre de métrica o dimensiones.

Note

Puede encontrar una lista de métricas y dimensiones publicadas por los servicios de AWS en [Referencia de dimensiones y métricas de Amazon CloudWatch](#) en la Guía del usuario de Amazon CloudWatch.

Importaciones

```
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);
}
}
```

Las métricas se devuelven en un [ListMetricsResponse](#) llamando a su `getMetrics` método.

Los resultados puede que estén paginados. Para recuperar el siguiente lote de resultados, llame a `nextToken` en el objeto de respuesta y use el valor de token para crear un nuevo objeto de solicitud. Llame entonces al método `listMetrics` de nuevo con la nueva solicitud.

Consulta el [ejemplo completo](#) en GitHub.

Más información

- [ListMetrics](#) en la referencia Amazon CloudWatch de la API

Publicar datos de métricas personalizadas para CloudWatch

Algunos servicios de AWS publican [sus propias métricas](#) en espacios de nombres que comienzan por "AWS". También puede publicar datos de métricas personalizadas usando su propio espacio de nombres (siempre y cuando no comience por "AWS").

Publicar datos de métricas personalizadas

Para publicar sus propios datos de métricas, llame al `putMetricData` método `CloudWatchClient`'s con un [PutMetricDataRequest](#). `PutMetricDataRequest` debe incluir el espacio de nombres personalizado que se utilizará para los datos e información sobre el propio punto de datos de un [MetricDatum](#) objeto.

Note

No puede especificar un espacio de nombres que comience por "AWS". Los espacios de nombres que comienzan por "AWS" están reservados para su uso por los productos de Amazon Web Services.

Importaciones

```
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);
}
```

Consulte el [ejemplo completo](#) en GitHub

Más información

- [Utilice Amazon CloudWatch las métricas](#) en la guía Amazon CloudWatch del usuario.
- [Espacios de nombres de AWS](#) en la Guía del usuario de Amazon CloudWatch.
- [PutMetricData](#) en la referencia Amazon CloudWatch de la API.

Uso de alarmas de CloudWatch

Crear una alarma

Para crear una alarma basada en una CloudWatch métrica, CloudWatchClient llama al putMetricAlarm método s con un [PutMetricAlarmRequest](#) relleno con las condiciones de la alarma.

Importaciones

```
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);
}
}
```

Consulta el [ejemplo completo](#) en GitHub.

Mostrar alarmas

Para ver una lista de las CloudWatch alarmas que ha creado, llame al CloudWatchClient describeAlarms método s con un [DescribeAlarmsRequest](#) que pueda usar para configurar las opciones del resultado.

Importaciones

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
```

Código

```
public static void desCWAlarms( CloudWatchClient cw) {
```



```
try {

    boolean done = false;
    String newToken = null;

    while(!done) {
        DescribeAlarmsResponse response;

        if (newToken == null) {
            DescribeAlarmsRequest request =
DescribeAlarmsRequest.builder().build();
            response = cw.describeAlarms(request);
        } else {
            DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
                .nextToken(newToken)
                .build();
            response = cw.describeAlarms(request);
        }

        for(MetricAlarm alarm : response.metricAlarms()) {
            System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
        }

        if(response.nextToken() == null) {
            done = true;
        } else {
            newToken = response.nextToken();
        }
    }

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

La lista de alarmas se puede obtener llamando `MetricAlarms` [DescribeAlarmsResponse](#) al devuelto por `describeAlarms`.

Los resultados puede que estén paginados. Para recuperar el siguiente lote de resultados, llame a `nextToken` en el objeto de respuesta y use el valor de token para crear un nuevo objeto de solicitud. Llame entonces al método `describeAlarms` de nuevo con la nueva solicitud.

Note

También puede recuperar las alarmas de una métrica específica mediante `CloudWatchClient` el `describeAlarmsForMetric` método. Su uso es similar a `describeAlarms`.

Consulte el [ejemplo completo](#) en GitHub.

Eliminación de alarmas

Para eliminar CloudWatch las alarmas, llame al `CloudWatchClient` `deleteAlarms` método [DeleteAlarmsRequest](#) que contenga uno o más nombres de las alarmas que desee eliminar.

Importaciones

```
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);
    }
}
```

Consulta el [ejemplo completo](#) en GitHub.

Más información

- [Uso de Amazon CloudWatch alarmas](#) en la Guía Amazon CloudWatch del usuario
- [PutMetricAlarm](#) en la referencia Amazon CloudWatch de la API
- [DescribeAlarms](#) en la referencia Amazon CloudWatch de la API
- [DeleteAlarms](#) en la referencia Amazon CloudWatch de la API

Usa Amazon CloudWatch Events

CloudWatch Los eventos ofrecen un flujo casi en tiempo real de los eventos del sistema que describen los cambios en AWS los recursos en las Amazon EC2 instancias, Lambda las funciones, los Kinesis flujos, Amazon ECS las tareas, las máquinas de Step Functions estado, Amazon SNS los temas, Amazon SQS las colas o los objetivos integrados. Mediante reglas sencillas, puede asignar los eventos y dirigirlos a una o más secuencias o funciones de destino.

Amazon EventBridge es la [evolución](#) de los CloudWatch eventos. Ambos servicios utilizan la misma API, por lo que puedes seguir utilizando el [cliente de CloudWatch Events](#) que proporciona el SDK o migrar al SDK del [EventBridge cliente](#) de Java para disfrutar de la funcionalidad de CloudWatch Events. CloudWatch [La documentación de la guía del usuario](#) de Events y la [referencia sobre la API](#) ya están disponibles en los sitios de EventBridge documentación.

Agregar eventos

Para añadir CloudWatch eventos personalizados, llama al `CloudWatchEventsClient`'s `putEvents` método con un [PutEventsRequest](#) objeto que contenga uno o más [PutEventsRequestEntry](#) objetos que proporcionen detalles sobre cada evento. Puede especificar varios parámetros para la entrada como el origen y el tipo del evento, los recursos asociados con el evento, etc.

Note

Puede especificar un máximo de 10 eventos para cada llamada a `putEvents`.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Agregar reglas

Para crear o actualizar una regla, llame al `CloudWatchEventsClient`'s `putRule` método con una [PutRuleRequest](#) con el nombre de la regla y parámetros opcionales, como el [patrón de eventos](#), el IAM rol que se va a asociar a la regla y una [expresión de programación](#) que describa la frecuencia con la que se ejecuta la regla.

Importaciones

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;
```

Código

```
public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {

    try {
        PutRuleRequest request = PutRuleRequest.builder()
            .name(ruleName)
            .roleArn(roleArn)
            .scheduleExpression("rate(5 minutes)")
            .state(RuleState.ENABLED)
            .build();

        PutRuleResponse response = cwe.putRule(request);
        System.out.printf(
            "Successfully created CloudWatch events rule %s with arn %s",
            roleArn, response.ruleArn());
    } catch (
        CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Agregar destinos

Los destinos son los recursos que se invocan cuando se activa una regla. Los objetivos de ejemplo incluyen Amazon EC2 instancias, Lambda funciones, Kinesis flujos, Amazon ECS tareas, máquinas de Step Functions estado y objetivos integrados.

Para añadir un objetivo a una regla, llama al `CloudWatchEventsClient`'s `putTargets` método con una [PutTargetsRequest](#) que contenga la regla que deseas actualizar y una lista de objetivos que añadir a la regla.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Más información

- [Añadir eventos PutEvents en la](#) Guía del EventBridge usuario de Amazon
- [Programa expresiones para reglas](#) en la Guía del EventBridge usuario de Amazon
- [Tipos de eventos para CloudWatch Events](#) la Guía del EventBridge usuario de Amazon

- [Patrones de eventos](#) en la guía del EventBridge usuario de Amazon
- [PutEvents](#) en la referencia de la EventBridge API de Amazon
- [PutTargets](#) en la referencia de la EventBridge API de Amazon
- [PutRule](#) en la referencia de la EventBridge API de Amazon

AWS servicios de bases de datos y AWS SDK for Java 2.x

AWS [ofrece varios tipos de bases de datos: relacionales, clave-valor, en memoria, de documentos y muchas más](#). La compatibilidad con el SDK for Java 2.x varía según la naturaleza del servicio de base de datos en AWS el que se encuentre.

Algunos servicios de bases de datos, por ejemplo el servicio [Amazon DynamoDB](#), tienen API de servicios web para administrar AWS el recurso (base de datos), así como API de servicios web para interactuar con los datos. En el SDK para Java 2.x, estos tipos de servicios tienen clientes de servicio dedicados, por ejemplo [DynamoDBClient](#).

Otros servicios de bases de datos tienen API de servicios web que interactúan con el recurso, como la API [Amazon DocumentDB](#) (para la administración de clústeres, instancias y recursos), pero no tienen una API de servicio web para trabajar con los datos. El SDK for Java 2.x tiene una [DocDbClient](#) interfaz correspondiente para trabajar con el recurso. Sin embargo, necesita otra API de Java, como [MongoDB para Java](#) para trabajar con los datos.

Utilice los ejemplos siguientes para aprender cómo utilizar los clientes de servicio SDK para Java 2.x con los distintos tipos de bases de datos.

Ejemplos de Amazon DynamoDB

Trabajo con los datos

Cliente de servicio SDK: [DynamoDbClient](#)

Ejemplo: aplicación [REST de React/Spring](#) con DynamoDB

Ejemplos: [varios ejemplos de DynamoDB](#)

Trabajo con la base de datos

Cliente de servicio SDK: [DynamoDbClient](#)

[CreateTableEjemplos ListTables:](#), DeleteTable

Trabajo con los datos

Cliente de servicio SDK: [DynamoDbEnhancedClient](#)

Ejemplo: aplicación [REST de React/Spring](#) con DynamoDB

Ejemplos: [varios ejemplos de DynamoDB \(nombres que comienzan por «Mejorado»\)](#)

Trabajo con la base de datos

Consulte [otros ejemplos de DynamoDB](#) en la sección de ejemplos de código guiados de esta guía.

Ejemplos de Amazon RDS

Trabajo con los datos	Trabajo con la base de datos
API ajena al SDK: JDBC, tipo SQL específico de la base de datos; el código administra las conexiones de la base de datos o un grupo de conexiones.	Cliente de servicio SDK: RdsClient
Ejemplo: aplicación REST de React/Spring con MySQL	Ejemplos: varios RdsClient ejemplos

Ejemplos de Amazon Redshift

Trabajo con los datos	Trabajo con la base de datos
Cliente de servicio SDK: RedshiftDataClient	Cliente de servicio SDK: RedshiftClient
Ejemplos: varios RedshiftDataClient ejemplos Ejemplo: aplicación REST de React/Spring que usa RedshiftDataClient	Ejemplos: varios RedshiftClient ejemplos

Ejemplos de Amazon Aurora Serverless v2

Trabajo con los datos	Trabajo con la base de datos
Cliente de servicio SDK: RdsDataClient	Cliente de servicio SDK: RdsClient
Ejemplo: aplicación REST de React/Spring que usa RdsDataClient	Ejemplos: varios ejemplos RdsClient

Ejemplos de Amazon DocumentDB

Trabajo con los datos	Trabajo con la base de datos
API que no pertenece al SDK: biblioteca Java específica de MongoDB (por ejemplo, MongoDB para Java); su código administra las conexiones de bases de datos o un grupo de conexiones.	Cliente de servicio SDK: DocDbClient
Ejemplos: Guía del desarrollador de DocumentDB (Mongo) (seleccione la pestaña «Java»)	

Trabaja con DynamoDB

Esta sección proporciona ejemplos que le muestran cómo trabajar con [DynamoDB](#).

Los ejemplos siguientes utilizan el cliente DynamoDB estándar de bajo nivel [DynamoDbClient](#) () de la versión 2.x. AWS SDK for Java

- [the section called “Trabaja con tablas en DynamoDB”](#)
- [the section called “Uso de elementos en DynamoDB”](#)

El SDK también ofrece el [cliente mejorado de DynamoDB](#) que proporciona un enfoque de alto nivel y orientado a objetos para trabajar con DynamoDB. En la siguiente sección, se analiza este cliente en profundidad.

- [the section called “Asignar objetos a elementos de DynamoDB”](#)

Trabaja con tablas en DynamoDB

Las tablas son los contenedores de todos los elementos de una DynamoDB base de datos. Para poder añadir o eliminar datos de DynamoDB ella, debe crear una tabla.

Para cada tabla, debe definir:

- Un nombre de tabla exclusivo para su cuenta y región.
- Una clave principal para la que cada valor debe ser único; no puede haber dos elementos de la tabla que tengan el mismo valor de clave principal.

Una clave principal puede ser simple, formada por una sola clave de partición (HASH) o compuesta, formada por una clave de partición y una clave de ordenación (RANGE).

Cada valor clave tiene un tipo de datos asociado, enumerado por la [ScalarAttributeType](#) clase. El valor de clave puede ser binario (B), numérico (N) o una cadena (S). Para obtener más información, consulte [Reglas de nomenclatura y tipos de datos](#) en la Guía para Amazon DynamoDB desarrolladores.

- El desempeño aprovisionado son valores que definen el número de unidades de capacidad de lectura/escritura reservadas para la tabla.

Note

[Amazon DynamoDB Los precios](#) se basan en los valores de rendimiento aprovisionados que estableces en tus tablas, así que reserva solo la capacidad que consideres que necesitarás para tu mesa.

El desempeño aprovisionado para una tabla se puede modificar en cualquier momento, por lo que puede ajustar la capacidad cuando cambien sus necesidades.

Creación de una tabla

Usa el `DynamoDbClient`'s `createTable` método para crear una tabla nueva DynamoDB . Debe crear los atributos de la tabla y un esquema de tabla, que se pueden usar para identificar la clave

principal de la tabla. También debe proporcionar los valores iniciales de desempeño aprovisionado y el nombre de una tabla.

Note

Si ya existe una tabla con el nombre que ha elegido, [DynamoDbException](#) se lanza una.

Crear una tabla con una clave principal simple

Este código crea una tabla con un atributo que es la clave principal simple de la tabla. En el ejemplo se utilizan [AttributeDefinition](#) [KeySchemaElement](#) objetos para la [CreateTableRequest](#).

Importaciones

```
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 "";
}

```

Consulte el [ejemplo completo](#) en GitHub

Crear una tabla con una clave primaria compuesta

En el siguiente ejemplo se crea una tabla con dos atributos. Ambos atributos se utilizan para la clave primaria compuesta.

Importaciones

```
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 "";
}
```

Consulte el [ejemplo completo](#) en GitHub.

Mostrar tablas

Puede enumerar las tablas de una región en particular llamando al `DynamoDbClient`'s `listTables` método.

Note

Si la tabla indicada no existe para tu cuenta y región, [ResourceNotFoundException](#) aparecerá una.

Importaciones

```
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!");
}
```

De forma predeterminada, se devuelven hasta 100 tablas por llamada; utilízalas `lastEvaluatedTableName` en el [ListTablesResponse](#) objeto devuelto para obtener la última tabla que se evaluó. Puede utilizar este valor para iniciar la enumeración después del último valor devuelto de la enumeración anterior.

Consulte el [ejemplo completo](#) en GitHub

Describir una tabla (obtener información de ella)

Utilice el método `describeTable` de `DynamoDbClient`'s para obtener información sobre una tabla.

Note

Si la tabla con el nombre indicado no existe para tu cuenta y región, [ResourceNotFoundException](#) aparecerá una.

Importaciones

```
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!");
}
```

Consulta el [ejemplo completo](#) en GitHub.

Modificar (actualizar) una tabla

Puede modificar los valores de rendimiento aprovisionado de la tabla en cualquier momento llamando al método `updateTable` de `DynamoDbClient`'s.

Note

Si la tabla con el nombre indicado no existe para tu cuenta y región, [ResourceNotFoundException](#) aparecerá una.

Importaciones

```
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!");
}
```

Consulta el [ejemplo completo](#) en GitHub.

Eliminación de una tabla

Para eliminar una tabla, llame al método `deleteTable` de `DynamoDbClient`'s y proporcione el nombre de la tabla.

Note

Si la tabla con el nombre indicado no existe para tu cuenta y región, [ResourceNotFoundException](#) aparecerá una.

Importaciones

```
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!");
}
```

Consulta el [ejemplo completo](#) en GitHub.

Más información

- [Prácticas recomendadas para trabajar con tablas](#) en la Guía para desarrolladores de Amazon DynamoDB
- [Cómo trabajar con tablas DynamoDB en](#) la guía para Amazon DynamoDB desarrolladores

Uso de elementos en DynamoDB

En DynamoDB, un elemento es una colección de atributos, cada uno de los cuales tiene un nombre y un valor. Los valores de los atributos pueden ser escalares, conjuntos o tipos de documentos. Para obtener más información, consulte [Reglas de nomenclatura y tipos de datos](#) en la Guía para desarrolladores de Amazon DynamoDB.

Recuperar (obtener) un elemento de una tabla

Llama al `getItem` método `DynamoDbClient`'s y pásale un [GetItemRequest](#) objeto con el nombre de la tabla y el valor de la clave principal del elemento que deseas. Devuelve un [GetItemResponse](#) objeto con todos los atributos de ese elemento. Puede especificar una o varias [expresiones de proyección](#) en `GetItemRequest` para recuperar atributos específicos.

Puede utilizar el `item()` método del `GetItemResponse` objeto devuelto para recuperar un [mapa](#) de los pares clave (cadena [AttributeValue](#)) y valor () asociados al elemento.

Importaciones

```
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);
    }
}
```

Consulta el [ejemplo completo](#) en GitHub.

Recuperar (obtener) un elemento de una tabla usando el cliente asíncrono

Invoke el `getItem` método del `DynamoDbAsyncClient` y pásale un [GetItemRequest](#) objeto con el nombre de la tabla y el valor de la clave principal del elemento que desee.

Puede devolver una instancia de [recopilación](#) con todos los atributos de ese elemento (consulte el siguiente ejemplo).

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub

Agregar un nuevo elemento a una tabla

Cree un [mapa](#) de pares de clave-valor que represente los atributos del elemento. Estos deben incluir valores para los campos de la clave principal de la tabla. Si el elemento identificado por la clave principal ya existe, la solicitud actualiza sus campos.

Note

Si la tabla con el nombre indicado no existe para tu cuenta y región, [ResourceNotFoundException](#) aparecerá una.

Importaciones

```
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);
    }
}
```

Consulta el [ejemplo completo](#) en GitHub.

Actualizar un elemento existente en una tabla

Puede actualizar un atributo de un elemento que ya existe en una tabla mediante el método `updateItem` de `DynamoDbClient`, proporcionando el nombre de la tabla, el valor de clave principal y un mapa de los campos que se van a actualizar.

Note

Si la tabla con el nombre asignado no existe para tu cuenta y región, o si el elemento identificado con la clave principal que has introducido no existe, [ResourceNotFoundException](#) aparecerá un.

Importaciones

```
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!");
}
```

Consulta el [ejemplo completo](#) en GitHub.

Eliminar un elemento existente en una tabla

Puede eliminar un elemento que existe en una tabla utilizando el `deleteItem` método `DynamoDbClient`'s y proporcionando un nombre de tabla, así como el valor de la clave principal.

Note

Si la tabla con el nombre asignado no existe para tu cuenta y región, o si el elemento identificado con la clave principal que has introducido no existe, [ResourceNotFoundException](#) se generará una.

Importaciones

```
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);
    }
}
```

Consulta el [ejemplo completo](#) en GitHub.

Más información

- [Prácticas recomendadas para el uso de elementos](#) en la Guía para desarrolladores de Amazon DynamoDB
- [Uso de elementos DynamoDB](#) en la Guía para desarrolladores de Amazon DynamoDB

Asignar objetos Java a elementos de DynamoDB con AWS SDK for Java 2.x

La [API de cliente mejorado de DynamoDB](#) es una biblioteca de alto nivel, sucesora de la clase `DynamoDBMapper` del SDK para Java v1.x. Ofrece una forma sencilla de asignar clases del cliente a tablas de DynamoDB. Las relaciones entre las tablas y sus clases de datos correspondientes se definen en el código. Después de definir estas relaciones, puede realizar intuitivamente varias operaciones de creación, lectura, actualización o eliminación (CRUD) en tablas o elementos en DynamoDB.

La API de cliente mejorado de DynamoDB también incluye [la API de documentos mejorada](#) para trabajar con elementos de tipo documento que no siguen un esquema definido.

La API de cliente mejorado de DynamoDB se analiza en los siguientes temas.

- [Empiece a utilizar la API de cliente mejorado de DynamoDB](#)
- [Aprenda los conceptos básicos de la API de cliente mejorado de DynamoDB](#)
- [Utilizar características de asignación avanzadas](#)
- [Trabaje con documentos JSON con la API de documentos mejorada para DynamoDB](#)
- [Usar extensiones](#)
- [Utilizar la API de cliente mejorado de DynamoDB de forma asíncrona](#)
- [Anotaciones de clases de datos](#)

Empiece a utilizar la API de cliente mejorado de DynamoDB

El siguiente tutorial presenta los aspectos básicos necesarios para trabajar con la API de cliente mejorado de DynamoDB.

agregar dependencias de API

Para empezar a trabajar con la API de cliente mejorado de DynamoDB en su proyecto, añada una dependencia al artefacto de Maven dynamodb-enhanced. Esto se muestra en el ejemplo siguiente.

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>
```

Haga una búsqueda de la [última versión](#) en el repositorio central de Maven y sustituya **<VERSION>** por este valor.

Gradle

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

```
implementation("software.amazon.awssdk:dynamodb-enhanced")
...
}
```

Haga una búsqueda de la [última versión](#) en el repositorio central de Maven y sustituya `<VERSION>` por este valor.

Generar a **TableSchema** partir de una clase de datos

Un [TableSchema](#) permite al cliente mejorado asignar valores de atributos de DynamoDB hacia y desde las clases del cliente. En este tutorial, conocerá los `TableSchema` derivados de una clase de datos estáticos y generadas a partir de código mediante un generador.

Utilice una clase de datos anotada

El SDK para Java 2.x incluye un [conjunto de anotaciones](#) que puede utilizar con una clase de datos para generar rápidamente un `TableSchema` para asignar sus clases a tablas.

Comience por crear una clase de datos que se ajuste a la [JavaBean especificación](#). La especificación requiere que una clase tenga un constructor público sin argumentos, además de getters y setters para cada atributo de la clase. Incluya una anotación a nivel de clase para indicar que la clase de datos es una `DynamoDbBean`. Además, como mínimo, incluya una anotación `DynamoDbPartitionKey` en el getter o setter para el atributo de clave principal.

Puede aplicar [anotaciones a nivel de atributo a](#) los captadores o a los definidores, pero no a ambos.

Note

El término `property` se utiliza normalmente para un valor encapsulado en un `JavaBean`. Sin embargo, en esta guía se utiliza el término `attribute` en su lugar para mantener la coherencia con la terminología utilizada por DynamoDB.

La siguiente clase `Customer` muestra las anotaciones que vinculan la definición de la clase a una tabla de DynamoDB.

Clase **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 + "];"
    }
}
```

Una vez creada una clase de datos anotada, utilízela para crear el `TableSchema`, como se muestra en el siguiente fragmento.

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

Un `TableSchema` se diseña para ser estático e inmutable. Por lo general, puede instanciarlo en el momento de cargar la clase.

El método de `TableSchema.fromBean()` fábrica estática realiza una introspección del bean para generar el mapeo de los atributos (propiedades) de las clases de datos hacia y desde los atributos de `DynamoDB`.

Para ver un ejemplo de cómo trabajar con un modelo de datos compuesto por varias clases de datos, consulte la clase `Person` en la sección [???](#).

Uso de un constructor

Puede saltarse el coste de la introspección de objetos bean si define el esquema de la tabla en código. Si codifica el esquema, no es necesario que la clase siga los estándares de `JavaBean` nomenclatura ni que esté anotada. El siguiente ejemplo utiliza un constructor y es equivalente al ejemplo de clase `Customer` que utiliza anotaciones.

```
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();
```

Crear un cliente mejorado y una **DynamoDbTable**

Crear un cliente mejorado

La [DynamoDbEnhancedClient](#) clase o su contraparte asíncrona [DynamoDbEnhancedAsyncClient](#), es el punto de partida para trabajar con la API de cliente mejorada de DynamoDB.

El cliente mejorado requiere un [DynamoDbClient](#) estándar para hacer el trabajo. La API ofrece dos formas de crear una instancia de [DynamoDbEnhancedClient](#). La primera opción, que se muestra en el siguiente fragmento, crea un [DynamoDbClient](#) estándar con los ajustes predeterminados tomados de la configuración.

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

Si desea configurar el cliente estándar subyacente, puede suministrarlo al método constructor del cliente mejorado como se muestra en el siguiente fragmento.

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

// Use the configured standard client with the enhanced client.
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
    .dynamoDbClient(standardClient)
    .build();
```

Crear una instancia de **DynamoDbTable**

Piense en una [DynamoDbTable](#) como la representación del cliente de una tabla de DynamoDB que utiliza la funcionalidad de mapeo proporcionada por un [TableSchema](#). La clase [DynamoDbTable](#) proporciona métodos para las operaciones CRUD que permiten interactuar con una sola tabla de DynamoDB.

[DynamoDbTable<T>](#) es una clase genérica que toma un único argumento de tipo, ya sea una clase personalizada o un [EnhancedDocument](#) cuando se trabaja con elementos de tipo documento. Este tipo de argumento establece la relación entre la clase que utiliza y la tabla única de DynamoDB.

Siga el método de fábrica `table()` del [DynamoDbEnhancedClient](#) para crear una instancia [DynamoDbTable](#), como se muestra en el siguiente fragmento.


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

Las instancias de `DynamoDbTable` son aptas para ser únicas porque son inmutables y se pueden usar en toda la aplicación.

El código ahora tiene una representación en memoria de una tabla de DynamoDB que puede funcionar con instancias. `Customer` La tabla de DynamoDB real puede existir o no. Si la tabla nombrada `Customer` ya existe, puede empezar a realizar operaciones de CRUD en ella. Si no existe, utilice la instancia de `DynamoDbTable` para crear la tabla, tal y como se explica en la siguiente sección.

Cree una tabla de DynamoDB si es necesario

Después de crear una instancia `DynamoDbTable`, úsela para crear una tabla de DynamoDB por única vez.

Crear código de ejemplo de tabla

En el siguiente ejemplo, se crea una tabla de DynamoDB basada en la clase de datos `Customer`.

En este ejemplo se crea una tabla DynamoDB con el nombre `Customer` —idéntico al nombre de la clase— pero el nombre de la tabla puede ser otro. Sea cual sea el nombre que dé a la tabla, debe usar este nombre en otras aplicaciones para trabajar con la tabla. Proporcione este nombre al método `table()` cada vez que cree otro objeto `DynamoDbTable` para trabajar con la tabla de DynamoDB subyacente.

El parámetro lambda de Java, `builder`, que se pasa al método `createTable`, permite [personalizar la tabla](#). En este ejemplo, se configura el [rendimiento aprovisionado](#). Si desea utilizar la configuración predeterminada al crear una tabla, omita el generador, tal y como se muestra en el siguiente fragmento.

```
customerTable.createTable();
```

Cuando se utiliza la configuración predeterminada, no se establecen los valores del rendimiento aprovisionado. En su lugar, el modo de facturación de la tabla se establece como [bajo demanda](#).

En el ejemplo también se utiliza una [DynamoDbWaiter](#) antes de intentar imprimir el nombre de la tabla recibido en la respuesta. Crear una tabla lleva algún tiempo. Por lo tanto, utilizar un 'waiter'

significa que no tiene que escribir lógica que consulte el servicio de DynamoDB para verificar si la tabla existe antes de su uso.

Importaciones

```
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> customerTable,
DynamoDbClient standardClient) {
    // Create the DynamoDB table using the 'customerTable' DynamoDbTable instance.
    customerTable.createTable(builder -> builder
        .provisionedThroughput(b -> b
            .readCapacityUnits(10L)
            .writeCapacityUnits(10L)
            .build())
    );
    // The DynamoDbClient instance (named 'standardClient') passed to the builder for
the DynamoDbWaiter is the same instance
    // that was passed to the builder of the DynamoDbEnhancedClient instance that we
created previously.
    // By using the same instance, it ensures that the same Region that was configured
on the standard DynamoDbClient
    // instance is used for other service clients that accept a DynamoDbClient during
construction.
    try (DynamoDbWaiter waiter =
DynamoDbWaiter.builder().client(standardClient).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

Los nombres de los atributos de una tabla de DynamoDB comienzan con una letra minúscula cuando la tabla se genera a partir de una clase de datos. Si desea que el nombre del atributo de la tabla comience con una letra mayúscula, utilice la [anotación `@DynamoDbAttribute\(NAME\)`](#) y proporcione el nombre que desee como parámetro.

Realizar operaciones

Una vez creada la tabla, utilice la instancia `DynamoDbTable` para las operaciones en la tabla de DynamoDB.

En el siguiente ejemplo, se pasa un singleton `DynamoDbTable<Customer>` como parámetro junto con una [instancia de clase de datos `Customer`](#) para añadir un nuevo elemento a la tabla.

```
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 el objeto `customer` al servicio DynamoDB, registre el resultado del método del objeto `toString()` para compararlo con lo que envía el cliente mejorado.

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

El registro a nivel de cable muestra la carga útil de la solicitud generada. El cliente mejorado generó la representación de bajo nivel a partir de la clase de datos. El atributo `regDate`, que es un tipo `Instant` en Java, se representa como una cadena de DynamoDB.

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

Trabajar con una tabla existente

En la sección anterior se muestra cómo crear una tabla de DynamoDB a partir de una clase de datos de Java. Si ya tiene una tabla existente y desea utilizar las características del cliente mejorado, puede crear una clase de datos de Java para que funcione con la tabla. Debe examinar la tabla de DynamoDB y añadir las anotaciones necesarias a la clase de datos.

Antes de trabajar con una tabla existente, llame al método `DynamoDbEnhanced.table()`. Esto se hizo en el ejemplo anterior con la instrucción siguiente.

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

Una vez devuelta la instancia `DynamoDbTable`, puede empezar a trabajar de inmediato con la tabla subyacente. No es necesario volver a crear la tabla llamando al método `DynamoDbTable.createTable()`.

El siguiente ejemplo lo demuestra mediante la recuperación inmediata de una instancia `Customer` de la tabla de 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

El nombre de la tabla utilizado en el método `table()` debe coincidir con el nombre de la tabla de DynamoDB existente.

Aprenda los conceptos básicos de la API de cliente mejorado de DynamoDB

Este tema trata las características básicas de la API de cliente mejorado de DynamoDB y la compara con la [API de cliente de DynamoDB estándar](#).

Si es la primera vez que utiliza la API de cliente mejorado de DynamoDB, le recomendamos que consulte el [tutorial introductorio](#) para familiarizarse con las clases fundamentales.

Elementos de DynamoDB en Java

Las tablas de DynamoDB almacenan elementos. Según el caso práctico, los elementos del lado de Java pueden adoptar la forma de datos estructurados de forma estática o estructuras creadas dinámicamente.

Si su caso requiere elementos con un conjunto coherente de atributos, utilice [clases anotadas](#) o un [constructor](#) para generar los tipos estáticos adecuados de `TableSchema`.

Alternativamente, si necesita almacenar elementos que consten de estructuras variables, cree un `DocumentTableSchema`. `DocumentTableSchema` forma parte de la [API de documento mejorada](#) y solo requiere una clave principal de tipo estático y funciona con instancias `EnhancedDocument` para contener los elementos de datos. La API de documentos mejorada se trata en otro [tema](#).

Tipos de atributos para clases de modelos de datos

Si bien DynamoDB admite [un número menor de tipos de atributos](#) en comparación con el sistema de tipos enriquecidos de Java, la API de cliente mejorado de DynamoDB proporciona mecanismos para convertir los miembros de una clase de Java a y desde tipos de atributos de DynamoDB.

Los tipos de atributos (propiedades) de las clases de datos de Java deben ser tipos de objetos, no primitivos. Por ejemplo, utilice siempre tipos Long de datos tipo Integer objeto, no Long int primitivos.

[De forma predeterminada, la API de cliente mejorada de DynamoDB admite convertidores de atributos para una gran cantidad de tipos, como Integer BigDecimal, String e Instant.](#) La lista aparece en las [clases de implementación conocidas de la AttributeConverter interfaz](#). La lista incluye muchos tipos y colecciones, como mapas, listas y conjuntos.

Para almacenar los datos de un tipo de atributo que no se admite de forma predeterminada o que no se ajusta a la JavaBean convención, puede escribir una AttributeConverter implementación personalizada para realizar la conversión. Consulte la sección sobre conversión de atributos para ver un [ejemplo](#).

Para almacenar los datos de un tipo de atributo cuya clase cumpla con la especificación de beans de Java (o una [clase de datos inmutable](#)), puede adoptar dos enfoques.

- Si tiene acceso al archivo fuente, puede anotar la clase con @DynamoDbBean (o @DynamoDbImmutable). La sección que analiza los atributos anidados muestra [ejemplos](#) del uso de clases anotadas.
- Si no tiene acceso al archivo de origen de la clase de JavaBean datos del atributo (o no quiere hacer anotaciones en el archivo de origen de una clase a la que sí tiene acceso), puede utilizar el enfoque de creación. Esto crea un esquema de tabla sin definir las claves. A continuación, puede anidar este esquema de tabla dentro de otro esquema de tabla para realizar el mapeo. La sección de atributos anidados contiene un [ejemplo](#) que muestra el uso de esquemas anidados.

Valores nulos

Al utilizar la API de putItem, el cliente mejorado no incluye los atributos con valores nulos de un objeto de datos mapeado en la solicitud a DynamoDB.

En el caso de las solicitudes de `updateItem`, los atributos con valores nulos se eliminan del elemento de la base de datos. Si va a actualizar algunos valores de atributo y mantener los demás sin cambios, copie los valores de otros atributos que no se deben cambiar o utilice el método [ignoreNull \(\)](#) del generador de actualizaciones.

El siguiente ejemplo demuestra `ignoreNulls()` para el 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 del cliente mejorado de DynamoDB

Los métodos básicos del cliente mejorado se corresponden con las operaciones del servicio de DynamoDB que les dan nombre. Los siguientes ejemplos muestran la variación más simple de cada método. Puede personalizar cada método pasando un objeto de solicitud mejorado. Los objetos de solicitud mejorados ofrecen la mayoría de las características disponibles en el cliente estándar de DynamoDB. Estas acciones se documentan por completo en la Referencia de la API de AWS SDK for Java 2.x .

El ejemplo usa la [the section called “Clase Customer”](#) mostrada 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 el cliente mejorado con el cliente estándar de DynamoDB

Las dos API de cliente de DynamoDB, [estándar](#) y [mejorada](#) permiten trabajar con tablas de DynamoDB para hacer operaciones CRUD (creación, lectura, actualización y eliminación) a nivel de datos. La diferencia entre las API de cliente reside en la forma en que se lleva a cabo. Con el cliente estándar, se trabaja directamente con atributos de datos de bajo nivel. La API de cliente mejorada utiliza clases de Java conocidas y se asigna a la API de bajo nivel en segundo plano.

Si bien ambas API de cliente admiten operaciones a nivel de datos, el cliente estándar de DynamoDB también admite operaciones a nivel de recursos. Las operaciones a nivel de recursos gestionan la base de datos, como la creación de copias de seguridad, la creación de listas y la actualización de tablas. La API de cliente mejorada admite un número selecto de operaciones a nivel de recursos, como la creación, descripción y eliminación de tablas.

Para ilustrar los diferentes enfoques utilizados por las dos API de cliente, los siguientes ejemplos de código muestran la creación de la misma tabla `ProductCatalog` con el cliente estándar y el cliente mejorado.

Comparación: crear una tabla mediante el cliente estándar de DynamoDB

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

Comparación: crear una tabla mediante el cliente mejorado de 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))
        )
    );
```

El cliente mejorado utiliza la siguiente clase de datos anotados. El cliente mejorado de DynamoDB asigna los tipos de datos de Java a los tipos de datos de DynamoDB para obtener un código menos detallado que sea más fácil de seguir. `ProductCatalog` es un ejemplo del uso de una clase inmutable con el cliente mejorado de DynamoDB. El uso de clases inmutables para las clases de datos mapeados [se analiza más adelante en este tema](#).

Clase `ProductCatalog`

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

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

import java.math.BigDecimal;
import java.util.Objects;
import java.util.Set;

@DynamoDbImmutable(builder = ProductCatalog.Builder.class)
public class ProductCatalog implements Comparable<ProductCatalog> {
    private Integer id;
    private String title;
    private String isbn;
    private Set<String> authors;
    private BigDecimal price;

    private ProductCatalog(Builder builder){
        this.authors = builder.authors;
        this.id = builder.id;
        this.isbn = builder.isbn;
        this.price = builder.price;
        this.title = builder.title;
    }
}
```

```
public static Builder builder(){ return new Builder(); }

@DynamoDbPartitionKey
public Integer id() { return id; }

@DynamoDbSortKey
public String title() { return title; }

@DynamoDbSecondaryPartitionKey(indexNames = "products_by_isbn")
public String isbn() { return isbn; }
public Set<String> authors() { return authors; }
public BigDecimal price() { return price; }

public static final class Builder {
    private Integer id;
    private String title;
    private String isbn;
    private Set<String> authors;
    private BigDecimal price;
    private Builder(){

    public Builder id(Integer id) { this.id = id; return this; }
    public Builder title(String title) { this.title = title; return this; }
    public Builder isbn(String ISBN) { this.isbn = ISBN; return this; }
    public Builder authors(Set<String> authors) { this.authors = authors; return
this; }
    public Builder price(BigDecimal price) { this.price = price; return this; }
    public ProductCatalog build() { return new ProductCatalog(this); }
}

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

```

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

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

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

```

Los dos ejemplos de código siguientes de una escritura por lotes ilustran la imprecisión y la falta de seguridad de tipos cuando se utiliza el cliente estándar frente al cliente mejorado.

Comparación: escritura por lotes mediante el cliente estándar de DynamoDB

```

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

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

    Map<String, AttributeValue> catalogItem2 = Map.of(
        "authors", AttributeValue.builder().ss("a", "b", "c").build(),

```

```

        "id", AttributeValue.builder().n("2").build(),
        "isbn", AttributeValue.builder().s("1-208-98073").build(),
        "title", AttributeValue.builder().s("Title 2").build(),
        "price", AttributeValue.builder().n("21.99").build());

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

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

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

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

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

```

Comparación: escritura por lotes mediante el cliente mejorado de 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());
}
```

Trabajar con clases de datos inmutables

La característica de asignación de la API de cliente mejorado de DynamoDB funciona con clases de datos inmutables. Una clase inmutable solo tiene getters y requiere una clase constructora que el SDK utiliza para crear instancias de la clase. En lugar de usar la anotación `@DynamoDbBean` como se muestra en la [clase Customer](#), las clases inmutables usan la anotación `@DynamoDbImmutable`, que toma un parámetro que indica la clase de generador que se va a usar.

La siguiente clase es una versión inmutable 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
        CustomerImmutable class.
        private Builder() {}

        public Builder id(String id) { 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); }
}
}
```

Debe cumplir los siguientes requisitos al anotar una clase de datos con `@DynamoDbImmutable`.

1. Todo método que no sea una sustitución de `Object.class` y que no haya sido anotado con `@DynamoDbIgnore` debe ser un método de obtención de un atributo de la tabla de DynamoDB.
2. Cada getter debe tener un setter correspondiente que distinga entre mayúsculas y minúsculas en la clase de constructor.
3. Solo debe cumplirse una de las siguientes condiciones de construcción.
 - La clase de constructor debe tener un constructor público predeterminado.
 - La clase de datos debe tener un nombre de método estático público `builder()` que no tome parámetros y devuelva una instancia de clase constructor. Esta opción se muestra en la clase `CustomerImmutable`.
4. La clase del constructor debe tener un método público llamado `build()` que no acepte parámetros y devuelva una instancia de la clase inmutable.

Para crear un `TableSchema` para su clase inmutable, utilice el método `fromImmutableClass()` en `TableSchema` como se muestra en el siguiente fragmento.

```
static final TableSchema<CustomerImmutable> customerImmutableTableSchema =
    TableSchema.fromImmutableClass(CustomerImmutable.class);
```

Del mismo modo que puede crear una tabla de DynamoDB a partir de una clase mutable, también puede crear una a partir de una clase inmutable con una llamada única a `createTable()` de `DynamoDbTable` como se muestra en el siguiente ejemplo de fragmento.

```
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));
}
```

Utilizar bibliotecas de terceros, como Lombok

Las bibliotecas de terceros, como [Project Lombok](#), ayudan a generar código reutilizable asociado a objetos inmutables. La API de cliente mejorado de DynamoDB funciona con estas bibliotecas siempre que las clases de datos sigan las convenciones detalladas en esta sección.

En el siguiente ejemplo, se muestra la clase `CustomerImmutable` inmutable con anotaciones de Lombok. Observe cómo la característica `onMethod` de Lombok copia las anotaciones de DynamoDB basadas en atributos, como `@DynamoDbPartitionKey`, en el código generado.

```
@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 expresiones y condiciones

Las expresiones de la API de cliente mejorado de DynamoDB son representaciones Java de las [expresiones de DynamoDB](#).

La API de cliente mejorado de DynamoDB utiliza tres tipos de expresiones:

[Expression](#)

La clase `Expression` se usa al definir condiciones y filtros.

[QueryConditional](#)

Este tipo de expresión representa las [condiciones clave para las](#) operaciones de consulta.

[UpdateExpression](#)

Esta clase le ayuda a escribir [expresiones de actualización](#) de DynamoDB y actualmente se usa en el marco de extensiones al actualizar un elemento.

Anatomía de una expresión

Una expresión consta de lo siguiente:

- Una expresión de cadena (obligatoria). La cadena contiene una expresión lógica de DynamoDB con nombres de marcadores de posición para los nombres y valores de los atributos.
- Un mapa de valores de la expresión (normalmente obligatorio).
- Un mapa de nombres de expresiones (opcional).

Utilice un creador para generar un objeto `Expression` que adopte la siguiente forma general.

```
Expression expression = Expression.builder()
    .expression(<String>)
    .expressionNames(<Map>)
    .expressionValues(<Map>)
    .build()
```

Las `Expression` suelen requerir un mapa de valores de expresión. El mapa proporciona los valores de los marcadores de posición de la expresión de cadena. La clave del mapa consta del nombre del marcador de posición precedido de dos puntos (`:`) y el valor del mapa es una instancia de [AttributeValue](#). La clase [AttributeValues](#) tiene métodos prácticos para generar una instancia de `AttributeValue` a partir de un literal. Como alternativa, puede usar el `AttributeValue.Builder` para generar una instancia de `AttributeValue`.

El siguiente fragmento muestra un mapa con dos entradas después de la línea de comentarios 2. La cadena pasada al método `expression()`, mostrada después de la línea de comentario 1,

contiene los marcadores de posición que DynamoDB resuelve antes de realizar la operación. Este fragmento no contiene un mapa de nombres de expresiones, ya que el precio es un nombre de atributo permitido.

```
public static void scanAsync(DynamoDbAsyncTable productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
            // provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
            // supplied as a map entry.
            .expressionValues(
                Map.of( ":min_value", numberValue(8.00),
                    ":max_value", numberValue(400_000.00)))
            .build())
        .build();
}
```

Si el nombre de un atributo de la tabla de DynamoDB es una palabra reservada, comienza por un número o contiene un espacio, se requiere un mapa de nombres de expresiones para el `Expression`.

Por ejemplo, si el nombre del atributo fuera *price* en lugar de *1price* en el ejemplo de código anterior, habría que modificar el ejemplo como se muestra en el siguiente ejemplo.

```
ScanEnhancedRequest request = ScanEnhancedRequest.builder()
    .filterExpression(Expression.builder()
        .expression("#price >= :min_value AND #price <= :max_value")
        .expressionNames( Map.of("#price", "1price") )
        .expressionValues(
            Map.of(":min_value", numberValue(8.00),
                ":max_value", numberValue(400_000.00)))
        .build())
    .build();
```

Un marcador de posición del nombre de una expresión comienza con el signo de almohadilla (#). Una entrada del mapa de nombres de expresiones utiliza el marcador de posición como clave y el nombre del atributo como valor. El mapa se agrega al creador de expresiones con el método `expressionNames()`. DynamoDB resuelve el nombre del atributo antes de efectuar la operación.

Los valores de expresión no son necesarios si se utiliza una función en la expresión de cadena. Un ejemplo de función de expresión es `attribute_exists(<attribute_name>)`.

En el ejemplo siguiente, se crea una `Expression` que utiliza una [función de DynamoDB](#). La cadena de expresión de este ejemplo no utiliza marcadores de posición. Esta expresión podría usarse en una operación `putItem` para comprobar si ya existe un elemento en la base de datos con un valor de atributo `movie` igual al atributo `movie` del objeto de datos.

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

La Guía para desarrolladores de DynamoDB contiene información completa sobre las [expresiones de bajo nivel](#) que se utilizan con DynamoDB.

Expresiones de condición y condicionales

Cuando se utilizan los métodos `putItem()`, `updateItem()` y `deleteItem()`, y también cuando se utilizan operaciones de transacción y lote, se usan objetos [Expression](#) para especificar las condiciones que DynamoDB debe cumplir para continuar con la operación. Estas expresiones se denominan expresiones de condición. Para ver un ejemplo, consulte la expresión de condición utilizada en el método `addDeleteItem()` (después de la línea de comentario 1) del [ejemplo de transacción](#) que se muestra en esta guía.

Cuando se trabaja con los métodos `query()`, una condición se expresa como [QueryConditional](#). La clase `QueryConditional` tiene varios métodos prácticos estáticos que le ayudan a escribir los criterios que determinan qué elementos leer de DynamoDB.

Para ver ejemplos de `QueryConditionals`, consulte el primer ejemplo de código de la sección [the section called “Ejemplos del método Query”](#) de esta guía.

Expresiones de filtro

Las expresiones de filtro se utilizan en las operaciones de análisis y consulta para filtrar los elementos que se devuelven.

Una expresión de filtro se aplica después de haber leído todos los datos de la base de datos, por lo que el coste de lectura es el mismo que si no hubiera filtro. La Guía para desarrolladores de Amazon DynamoDB contiene más información sobre el uso de expresiones de filtro para las operaciones de [consulta](#) y [análisis](#).

En el ejemplo siguiente, se muestra una expresión de filtro agregada a una solicitud de análisis. El criterio restringe los artículos devueltos a artículos con un precio entre 8 y 80 euros, ambos incluidos.

```
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();
```

Expresiones de actualización

El método `updateItem()` de cliente mejorado de DynamoDB proporciona una forma estándar de actualizar elementos en DynamoDB. Sin embargo, cuando necesite más funciones, [UpdateExpressions](#) proporciona una representación segura de la [sintaxis de las expresiones de actualización](#) de DynamoDB. Por ejemplo, puede usar `UpdateExpressions` para aumentar los valores sin leer primero los elementos de DynamoDB o agregar miembros individuales a una lista. Las expresiones de actualización están disponibles actualmente en extensiones personalizadas para el método `updateItem()`.

Para ver un ejemplo en el que se utilizan expresiones de actualización, consulte el [ejemplo de extensión personalizada](#) de esta guía.

Encontrará más información sobre las expresiones de actualización en la [Guía para desarrolladores de Amazon DynamoDB](#).

Trabajar con resultados paginados: análisis y consultas

Los métodos `scan`, `query` y `batch` de la API de cliente mejorado de DynamoDB devuelven respuestas con una o varias páginas. Una página contiene uno o varios elementos. El código puede procesar la respuesta por página o procesar elementos individuales.

Una respuesta paginada devuelta por el `DynamoDbEnhancedClient` cliente síncrono devuelve un [PageIterable](#) objeto, mientras que una respuesta devuelta por el cliente asíncrono `DynamoDbEnhancedAsyncClient` devuelve un objeto. [PagePublisher](#)

En esta sección se analiza el procesamiento de los resultados paginados y se proporcionan ejemplos en los que se utilizan las API de análisis y consulta.

Examinar una tabla

El método [scan](#) del SDK corresponde a la [operación de DynamoDB](#) del mismo nombre. La API de cliente mejorado de DynamoDB ofrece las mismas opciones, pero utiliza un modelo de objetos conocido y gestiona la paginación por usted.

En primer lugar, exploramos la `PageIterable` interfaz analizando el `scan` método de la clase de mapeo síncrono, [DynamoDbTable](#)

Utilizar la API síncrona

El siguiente ejemplo muestra el método `scan` que usa una [expresión](#) para filtrar los elementos que se devuelven. [ProductCatalog](#) Es el objeto modelo que se mostró anteriormente.

La expresión de filtrado que se muestra después de la línea de comentario 2 limita los `ProductCatalog` artículos devueltos a aquellos con un precio comprendido entre 8,00 y 80,00€, ambos inclusive.

En este ejemplo también se excluyen `isbn` los valores mediante el `attributesToProject` método que se muestra después de la línea de comentario 1.

Tras la línea de comentario 3 `pagedResults`, el `scan` método devuelve el `PageIterable` objeto. El método `stream` de `PageIterable` devuelve un objeto [java.util.Stream](#), que puede utilizar para procesar las páginas. En este ejemplo, se cuenta y se registra el número de páginas.

Empezando por la línea de comentarios 4, el ejemplo muestra dos variantes del acceso a los elementos de `ProductCatalog`. La versión posterior a la línea de comentarios 4a recorre cada página y ordena y registra los elementos de cada página. La versión siguiente a la línea de comentarios 4b omite la iteración de la página y accede a los elementos directamente.

La interfaz `PageIterable` ofrece varias formas de procesar los resultados debido a sus dos interfaces principales, [java.lang.Iterable](#) y [SdkIterable](#). `Iterable` trae los métodos `forEach`, `iterator` y `splititerator`, y `SdkIterable` el 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())
        );
}

```

Utilizar la API asíncrona

El método `scan` asíncrono devuelve los resultados como un objeto `PagePublisher`. La interfaz de `PagePublisher` tiene dos métodos `subscribe` que puede utilizar para procesar las páginas de respuesta. Un método `subscribe` proviene de la interfaz principal

`org.reactivestreams.Publisher`. Para procesar páginas con esta primera opción, pase una instancia [Subscriber](#) al método `subscribe`. En el ejemplo siguiente se muestra el uso del método `subscribe`.

El segundo `subscribe` método proviene de la interfaz [SdkPublisher](#). Esta versión de `subscribe` acepta un [Consumer](#) en lugar de un `Subscriber`. Esta variación del método `subscribe` se muestra en el segundo ejemplo siguiente.

El ejemplo siguiente muestra la versión asíncrona del método `scan` que utiliza la misma expresión de filtro que se muestra en el ejemplo anterior.

Tras la línea de comentario 3, `DynamoDbAsyncTable.scan` devuelve un objeto `PagePublisher`. En la siguiente línea, el código crea una instancia de la interfaz de `org.reactivestreams.Subscriber`, `ProductCatalogSubscriber`, que se suscribe a la `PagePublisher` después de la línea de comentarios 4.

El objeto `Subscriber` recopila los elementos `ProductCatalog` de cada página del método `onNext` después de la línea de comentarios 8 del ejemplo de clase `ProductCatalogSubscriber`. Los elementos se almacenan en la variable `List` privada y se accede a ellos en el código de llamada con el método `ProductCatalogSubscriber.getSubscribedItems()`. Esto se invoca después de la línea de comentarios 5.

Una vez recuperada la lista, el código ordena todos los artículos `ProductCatalog` por precio y registra cada uno de ellos.

El comando [CountDownLatch](#) in the `ProductCatalogSubscriber` class bloquea el hilo de llamada hasta que todos los elementos se hayan agregado a la lista antes de continuar después de la línea de comentarios 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;
    }
}

```

En el siguiente ejemplo de fragmento, se utiliza la versión del método `PagePublisher.subscribe` que acepta una `Consumer` después de la línea de comentario 6. El parámetro `lambda` de Java consume páginas, que procesan aún más cada elemento. En este ejemplo, se procesa cada página y los elementos de cada página se ordenan y, a continuación, se registran.

```

// 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.
```

El método `items` de `PagePublisher` separa las instancias del modelo para que el código pueda procesar los elementos directamente. Este método se muestra en el fragmento de código siguiente.

```
// 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 una tabla

El método [`query\(\)`](#) de la clase `DynamoDbTable` busca elementos según los valores de clave principal. La anotación `@DynamoDbPartitionKey` y la anotación opcional `@DynamoDbSortKey` se utilizan para definir la clave principal de la clase de datos.

El método `query()` requiere un valor de clave de partición que busque los elementos que coincidan con el valor proporcionado. Si su tabla también define una clave de clasificación, puede añadir un valor de la misma a su consulta como condición de comparación adicional para afinar los resultados.

Excepto por el procesamiento de los resultados, las versiones síncrona y asíncrona de `query()` funcionan igual. Igual que con la API de `scan`, la API de `query` devuelve una `PageIterable` para una llamada síncrona y una `PagePublisher` para una llamada asíncrona. Hemos discutido el uso de `PageIterable` y `PagePublisher` anteriormente en la sección de análisis.

Ejemplos del método `Query`

El ejemplo de código del método `query()` siguiente utiliza la clase `MovieActor`. La clase de datos define una clave primaria compuesta que se compone del atributo `movie` de la clave de partición y el atributo `actor` de la clave de clasificación.

La clase también indica que utiliza un índice secundario global denominado `acting_award_year`. La clave primaria compuesta del índice se compone del atributo `actingaward` de la clave de partición y el `actingyear` de la clave de clasificación. Más adelante en este tema, cuando mostremos cómo crear y usar índices, nos referiremos al índice `acting_award_year`.

Clase **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);
        }
    }
}

```

Los ejemplos de código que aparecen a continuación se refieren a los siguientes elementos.

Elementos de la tabla **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'}

```

El siguiente código define dos [QueryConditional](#) instancias. [QueryConditional](#)s funcionan con valores clave (ya sea la clave de partición sola o en combinación con la clave de ordenación) y corresponden a las [expresiones condicionales clave de la API del servicio](#) DynamoDB. Tras la línea de comentarios 1, en el ejemplo se define la instancia `keyEqual` que hace coincidir los elementos con un valor de partición de **movie01**.

Este ejemplo también define una expresión de filtro que filtra cualquier elemento que no tenga **actingschoolname** después de la línea de comentario 2.

Tras la línea de comentarios 3, en el ejemplo se muestra la [QueryEnhancedRequest](#) instancia en la que el código pasa al método `DynamoDbTable.query()`. Este objeto combina la condición clave y el filtro que utiliza el SDK para generar la solicitud al servicio 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.
    );

```

Se genera la siguiente salida de la ejecución del método. El resultado muestra los elementos con un valor `movieName` de `movie01` y no muestra ningún elemento con `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'}

```

En la siguiente variación de solicitud de consulta mostrada anteriormente tras la línea de comentario 3, el código sustituye el `keyEqual` `QueryConditional` por el `sortGreaterThanOrEqualTo` `QueryConditional` que se definió tras la línea de comentario 1a. El código siguiente también elimina la expresión del filtro.

```
QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
```

```
.queryConditional(sortGreaterThanOrEqualTo)
```

Dado que esta tabla tiene una clave primaria compuesta, todas las instancias de `QueryConditional` requieren un valor de clave de separación. Los métodos `QueryConditional` que comienzan por `sort...` indican que se requiere una clave de clasificación. Los resultados no están ordenados.

La siguiente salida muestra los resultados de la consulta. La consulta devuelve los elementos que tienen un valor `movieName` igual a `movie01` y solo los elementos que tienen un valor `actorName` mayor o igual a `actor2`. Como se quitó el filtro, la consulta devuelve los elementos que no tienen ningún valor para el 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 operaciones por lotes

La API de cliente mejorado de DynamoDB ofrece dos métodos por lotes, [batchGetItem\(\)](#) y [batchWriteItem\(\)](#).

`batchGetItem()` Ejemplo de

Con el método [DynamoDbEnhancedClient.batchGetItem\(\)](#), puede recuperar hasta 100 elementos individuales de varias tablas en una solicitud general. En el siguiente ejemplo, se utilizan las clases de datos [Customer](#) y [MovieActor](#) mostradas anteriormente.

En el ejemplo, después de las líneas 1 y 2, se crean objetos [ReadBatch](#) que se añaden posteriormente como parámetros al método `batchGetItem()` después de la línea de comentarios 3. El código que sigue a la línea de comentarios 1 crea el lote para leerlo de la tabla `Customer`. El código que sigue a la línea de comentarios 1a muestra el uso de un generador [GetItemEnhancedRequest](#) que toma los valores de la clave principal para especificar el elemento que se va a leer. A diferencia de especificar valores clave para solicitar un elemento, puede usar una

clase de datos para solicitar un elemento, como se muestra después de la línea de comentario 1b. El SDK extrae automáticamente los valores clave de manera interna antes de enviar la solicitud.

Cuando especifique el elemento utilizando el enfoque basado en claves, como se muestra en las dos instrucciones posteriores a 2a, también puede especificar que DynamoDB realice una [lectura altamente coherente](#). Cuando se utilice el método `consistentRead()`, se debe utilizar en todos los elementos solicitados de la misma tabla.

Para recuperar los elementos encontrados por DynamoDB, utilice el método [resultsForTable\(\)](#) que se muestra después de la línea de comentarios 4. Llame al método de cada tabla que se haya leído en la solicitud. `resultsForTable()` devuelve una lista de los elementos encontrados que puede procesar mediante cualquier método `java.util.List`. En este ejemplo se registra cada elemento.

Para descubrir elementos que DynamoDB no procesó, utilice el enfoque que aparece después de la línea de comentarios 5. La clase `BatchGetResultPage` tiene el método [unprocessedKeysForTable\(\)](#) que permite acceder a todas las claves que no se hayan procesado. La [referencia de la BatchGetItem API](#) contiene más información sobre situaciones que dan como resultado elementos sin procesar.

```
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.
        .addGetItem(b -> b.key(k ->
k.partitionValue("1").sortValue("cust1@orgname.org")))
        // 1b. Alternatively, supply a data class instances to provide the
primary key values.
        .addGetItem(customer2)
        .build();

    // 2. Build a batch to read from the MovieActor table.
    ReadBatch movieActorBatch = ReadBatch.builder(MovieActor.class)
        .mappedTableResource(movieActorTable)
```

```

        // 2a. Call consistentRead(Boolean.TRUE) for each item for the same
table.
        .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor1")).consistentRead(Boolean.TRUE))
        .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor4")).consistentRead(Boolean.TRUE))
        .build();

        // 3. Add ReadBatch objects to the request.
        BatchGetResultPageIterable resultPages = enhancedClient.batchGetItem(b ->
b.readBatches(customerBatch, moveActorBatch));

        // 4. Retrieve the successfully requested items from each table.
        resultPages.resultsForTable(customerTable).forEach(item ->
logger.info(item.toString()));
        resultPages.resultsForTable(movieActorTable).forEach(item ->
logger.info(item.toString()));

        // 5. Retrieve the keys of the items requested but not processed by the
service.
        resultPages.forEach((BatchGetResultPage pageResult) -> {
            pageResult.unprocessedKeysForTable(customerTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
            pageResult.unprocessedKeysForTable(movieActorTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
        });
    }
}

```

Suponga que los siguientes elementos están en las dos tablas antes de ejecutar el código de ejemplo.

Elementos de la tabla

```

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'}

```

El siguiente resultado muestra los elementos devueltos y registrados después de la línea de comentarios 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'}

```

batchWriteItem() Ejemplo de

El método `batchWriteItem()` pone o borra varios elementos en una o varias tablas. Puede especificar hasta 25 operaciones individuales de colocación o borrado en la solicitud. En el siguiente ejemplo, se utilizan las clases de modelos [ProductCatalog](#) y [MovieActor](#) mostradas anteriormente.

Los objetos de `WriteBatch` se crean después de las líneas de comentario 1 y 2. En la tabla `ProductCatalog`, el código coloca un elemento y elimina otro. En la tabla `MovieActor` situada después de la línea de comentarios 2, el código coloca dos elementos y elimina uno.

El método `batchWriteItem` se llama después de la línea de comentarios 3. El parámetro [builder](#) proporciona las solicitudes de lote para cada tabla.

El objeto devuelto [BatchWriteResult](#) proporciona métodos independientes para cada operación a fin de ver las solicitudes no procesadas. El código que sigue a la línea de comentarios 4a proporciona las claves para las solicitudes de eliminación sin procesar y el código que sigue a la línea de comentarios 4b proporciona los elementos de venta sin procesar.

```

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()));
    }
}

```

```
}
```

Los siguientes métodos auxiliares proporcionan los objetos modelo para las operaciones de colocación y eliminación.

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;
}
```

Suponga que las tablas contienen los siguientes elementos antes de ejecutar el código de ejemplo.

```
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}
```

Una vez finalizado el código de ejemplo, las tablas contienen los siguientes elementos.

```
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 en la tabla `MovieActor` que el elemento de la película `Blue Jasmine` se ha sustituido por el elemento utilizado en la solicitud de venta adquirida mediante el método auxiliar `getMovieActorBlanchettPartial()`. Si no se ha proporcionado el valor de un atributo de un objeto de datos, se elimina el valor de la base de datos. Esta es la razón por la que el `actingSchoolName` resultante es nulo para el elemento de la película `Blue Jasmine`.

Note

Aunque la documentación de la API sugiere que pueden utilizarse expresiones de condición y que las métricas de capacidad consumida y de recogida pueden devolverse con solicitudes individuales de [put](#) y [delete](#), esto no es así en un escenario de escritura por lotes. Para mejorar el rendimiento de las operaciones por lotes, se ignoran estas opciones individuales.

Realizar operaciones de transacciones

La API de cliente mejorado de DynamoDB proporciona los `transactGetItems()` y los métodos `transactWriteItems()`. Los métodos de transacción del SDK para Java proporcionan atomicidad, consistencia, aislamiento y durabilidad (ACID) en las tablas de DynamoDB, ayudándole a mantener la corrección de los datos en sus aplicaciones.

`transactGetItems()` Ejemplo de

El método [transactGetItems\(\)](#) admite hasta 100 solicitudes individuales de artículos. Todos los artículos se leen en una sola transacción atómica. La Guía para desarrolladores de Amazon DynamoDB contiene información sobre las [condiciones que hacen que un método transactGetItems\(\) falle](#), y también sobre el nivel de aislamiento utilizado cuando se llama a [transactGetItem\(\)](#).

Tras la línea de comentarios 1 del siguiente ejemplo, el código llama al método `transactGetItems()` con un parámetro [builder](#). El generador [addGetItem\(\)](#) se invoca tres veces con un objeto de datos que contiene los valores clave que el SDK utilizará para generar la solicitud final.

La solicitud devuelve una lista de objetos [Document](#) después de la línea de comentarios 2. La lista de documentos que se devuelve contiene instancias de [documentos](#) no nulas de los datos de los artículos en el mismo orden solicitado. El método [Document.getItem\(MappedTableResource<T> mappedTableResource\)](#) convierte un objeto sin tipo en un objeto `Document` Java con tipo si se devolvieron los datos del elemento; de lo contrario, el método devuelve un valor 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());
}
}
```

Las tablas de DynamoDB contienen los siguientes elementos antes de que se ejecute el ejemplo 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'}
```

Se registra la siguiente salida. Si se solicita un elemento pero no se encuentra, no se devuelve, como ocurre con la solicitud de la película nombrada 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'}
```

Ejemplos de `transactWriteItems()`

El [transactWriteItems\(\)](#) acepta hasta 100 acciones para colocar, actualizar o eliminar en una única transacción atómica a través de varias tablas. La Guía para desarrolladores de Amazon DynamoDB contiene detalles sobre las restricciones y las condiciones de fallo del [funcionamiento del servicio DynamoDB subyacente](#).

Ejemplo básico

En el ejemplo siguiente, se solicitan cuatro operaciones para dos tablas. Las clases de modelos correspondientes [ProductCatalog](#) y [MovieActor](#) se mostraron anteriormente.

Cada una de las tres operaciones posibles (poner, actualizar y eliminar) utiliza un parámetro de solicitud específico para especificar los detalles.

El código bajo la línea de comentarios 1 muestra la variación simple del método `addPutItem()`. El método acepta un objeto [MappedTableResource](#) y la instancia del objeto de datos para colocarlo. La declaración que sigue a la línea de comentarios 2 muestra la variación que acepta una instancia [TransactPutItemEnhancedRequest](#). Esta variante permite añadir más opciones a la solicitud, como una expresión de condición. En un [ejemplo](#) posterior, se muestra una expresión de condición para una operación individual.

Se solicita una operación de actualización después de la línea de comentario 3.

[TransactUpdateItemEnhancedRequest](#) tiene un método `ignoreNulls()` que permite configurar lo que hace el SDK con los valores `null` del objeto modelo. Si el método `ignoreNulls()` devuelve el valor `true`, el SDK no elimina los valores de los atributos de la tabla para los atributos del objeto de datos que son `null`. Si el método `ignoreNulls()` devuelve `false`, el SDK solicita al servicio DynamoDB que elimine los atributos del elemento de la tabla. El valor predeterminado de `ignoreNulls` es `false`.

La instrucción tras la línea de comentario 4 muestra la variación de una petición de borrado que toma un objeto de datos. El cliente mejorado extrae los valores clave antes de enviar la solicitud 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())
    );
}

```

Los siguientes métodos auxiliares proporcionan los objetos de datos para los parámetros 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;
}

```

Las tablas de DynamoDB contienen los siguientes elementos antes de que se ejecute el ejemplo 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'}

```

Los siguientes elementos aparecen en las tablas después de que el código termine de ejecutarse.

```

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'}

```

Se ha eliminado el elemento de la línea 2 y las líneas 3 y 5 muestran los elementos que se colocaron. La línea 4 muestra la actualización de la línea 1. El valor `price` es el único que ha cambiado en el artículo. Si `ignoreNulls()` hubiera devuelto `false`, la línea 4 tendría el siguiente aspecto.

```
ProductCatalog{id=4, title='Title 1', isbn='null', authors=null, price=40.0}
```

Ejemplo de comprobación de estado

En el siguiente ejemplo, se muestra el uso de una comprobación de estado. Una comprobación de estado se utiliza para comprobar la existencia de un elemento o para comprobar el estado de atributos concretos de un elemento en la base de datos. El artículo registrado en la verificación de estado no se puede utilizar en otra operación de la transacción.

Note

No se pueden dirigir varias operaciones de la misma transacción al mismo elemento. Por ejemplo, no puede llevar a cabo una comprobación de condiciones y además intentar actualizar el mismo artículo en la misma transacción.

El ejemplo muestra una operación de cada tipo en una solicitud transaccional de escritura de artículos. Después de la línea de comentario 2, el método `addConditionCheck()` suministra la condición que falla la transacción si el parámetro `conditionExpression` se evalúa como `false`. La expresión de condición que devuelve el método mostrado en el bloque Métodos auxiliares comprueba si el año de concesión de la película *Sophie's Choice* no es igual a 1982. Si es así, la expresión se evalúa como `false` y la transacción no se realiza correctamente.

Esta guía analiza en profundidad [las expresiones](#) en otro tema.

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

```

En el ejemplo de código anterior se utilizan los siguientes métodos auxiliares.

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;
}

```

Las tablas de DynamoDB contienen los siguientes elementos antes de que se ejecute el ejemplo 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'}

```

Los siguientes elementos aparecen en las tablas después de que el código termine de ejecutarse.

```

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'}

```

Los elementos permanecen sin cambios en las tablas porque se produjo un error en la transacción. El valor `actingYear` de la película `Sophie's Choice` es 1982, como muestra la línea 2 de los elementos de la tabla antes de llamar al método `transactWriteItem()`.

Para capturar la información de cancelación de la transacción, incluya la llamada al método `transactWriteItems()` en un bloque `try` y `catch` el [TransactionCanceledException](#). Tras

la línea de comentario 4 del ejemplo, el código registra cada objeto [CancellationReason](#). Dado que el código que sigue a la línea de comentario 3 del ejemplo especifica que deben devolverse los valores del elemento que provocó el fallo de la transacción, el registro muestra los valores brutos de la base de datos para el elemento de la película 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.)
```

Ejemplo de condición de operación única

El siguiente ejemplo muestra el uso de una condición en una sola operación de una solicitud de transacción. La operación de eliminación después de la línea de comentarios 1 contiene una condición que compara el valor del elemento objetivo de la operación con la base de datos. En este ejemplo, la expresión de condición creada con el método auxiliar después de la línea de comentario 2 especifica que el elemento debe eliminarse de la base de datos si el año de actuación de la película no es igual a 2013.

Las [expresiones](#) se describen más adelante en esta guía.

```
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();
}
}

```

En el ejemplo de código anterior se utilizan los siguientes métodos auxiliares.

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;
}

```

Las tablas de DynamoDB contienen los siguientes elementos antes de que se ejecute el ejemplo 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'}

```

Los siguientes elementos aparecen en las tablas después de que el código termine de ejecutarse.

```

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'}

```

Los elementos permanecen sin cambios en las tablas porque se produjo un error en la transacción. El valor `actingYear` para la película `Blue Jasmine` es `2013` como se muestra en la línea 2 de la lista de elementos antes de que se ejecute el ejemplo de código.

Las líneas siguientes se registran en la consola.

```

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 secundarios

Los índices secundarios mejoran el acceso a los datos al definir claves alternativas que se utilizan en las operaciones de consulta y análisis. Índice secundario global (GSI): índice con una clave de partición y una clave de clasificación que pueden ser diferentes de las de la tabla base. Por el contrario, los índices secundarios locales (LSI) utilizan la clave de partición del índice principal.

Anote la clase de datos con anotaciones de índice secundarias

Los atributos que participan en índices secundarios requieren la anotación `@DynamoDbSecondaryPartitionKey` o `@DynamoDbSecondarySortKey`.

La siguiente clase muestra las anotaciones de dos índices. El GSI denominado `SubjectLastPostedDateIndex` usa el `Subject` atributo para la clave de partición y el atributo `LastPostedDateTime` para la clave de clasificación. El LSI denominado `ForumLastPostedDateIndex` utiliza el `ForumName` como clave de partición y `LastPostedDateTime` como clave de clasificación.

Tenga en cuenta que el atributo `Subject` cumple una doble función. Es la clave de clasificación de la clave principal y la clave de partición del GSI denominado. `SubjectLastPostedDateIndex`

Clase `MessageThread`

La clase `MessageThread` es adecuada para utilizarla como clase de datos en la [tabla Thread de ejemplo](#) de la Guía para desarrolladores de Amazon DynamoDB.

Importaciones

```
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 +
        '}';
}
}

```

Crear el índice

A partir de la versión 2.20.86 del SDK para Java, el método `createTable()` genera automáticamente índices secundarios a partir de las anotaciones de las clases de datos. De forma predeterminada, todos los atributos de la tabla base se copian en un índice y los valores de rendimiento aprovisionados son 20 unidades de capacidad de lectura y 20 unidades de capacidad de escritura.

Sin embargo, si utiliza una versión del SDK anterior a la 2.20.86, debe crear el índice junto con la tabla, como se muestra en el siguiente ejemplo. En este ejemplo, se crean los dos índices de la tabla Thread. El parámetro [constructor](#) tiene métodos para configurar ambos tipos de índices, como se muestra después de las líneas de comentario 1 y 2. Utilice el método del generador de índices `indexName()` para asociar los nombres de índice especificados en las anotaciones de las clases de datos con el tipo de índice deseado.

Este código configura todos los atributos de la tabla para que terminen en ambos índices después de las líneas de comentario 3 y 4. Encontrará más información sobre las [proyecciones de atributos](#) en la Guía para desarrolladores de 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 utilizando un índice

En el siguiente ejemplo, se consulta el índice `ForumLastPostedDateIndex` secundario local.

Tras la línea de comentario 2, se crea un [QueryConditional](#) objeto que es obligatorio al llamar al método [DynamoDbIndex.query\(\)](#).

Para obtener una referencia al índice que desea consultar después de la línea de comentario 3, debe pasar al nombre del índice. Tras la línea de comentario 4, se llama al método `query()` del índice que pasa al objeto `QueryConditional`.

También puede configurar la consulta para que devuelva tres valores de atributo, como se muestra tras la línea de comentario 5. Si no se llama `attributesToProject()`, la consulta devuelve

todos los valores de los atributos. Los nombres de los atributos especificados comienzan con letras minúsculas. Estos nombres coinciden con los utilizados en la tabla, no necesariamente con los de la clase de datos.

Siguiendo la línea de comentario 6, se itera a través de los resultados y se registra cada elemento devuelto por la consulta y también se almacena en la lista para devolvérsela a la persona que llama.

```
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;
}

```

En la base de datos existen los siguientes elementos antes de que se ejecute la 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}

```

Las instrucciones de registro de las líneas 1 y 6 dan como resultado la siguiente salida de consola.

```

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}
```

La consulta devolvió elementos con un valor `forumName` de `Forum02` y un valor de `lastPostedDateTime` superior o igual a `2023.03.31`. Los resultados muestran valores `message` con una cadena vacía, aunque los atributos `message` tienen valores en el índice. Esto se debe a que el atributo del mensaje no se proyectó después de la línea de comentario 5.

Utilizar características de asignación avanzadas

Obtenga información sobre las características avanzadas del esquema de tablas en la API del cliente mejorado de DynamoDB.

Comprender los tipos de esquema de tabla

[TableSchema](#) es la interfaz para la funcionalidad de asignación de la API del cliente mejorado de DynamoDB. Puede mapear un objeto de datos hacia y desde un mapa de [AttributeValues](#). Un objeto `TableSchema` necesita conocer la estructura de la tabla que está asignando. Esta información de estructura se almacena en un objeto [TableMetadata](#).

La API del cliente mejorado tiene varias implementaciones de `TableSchema`, indicadas a continuación.

Esquema de tabla generado a partir de clases anotadas

Es una operación algo costosa construir un `TableSchema` a partir de clases anotadas, por lo que recomendamos hacerlo una vez, al inicio de la aplicación.

[BeanTableSchema](#)

Esta implementación se construye basándose en atributos y anotaciones de una clase de bean. En la sección [Introducción](#) se muestra un ejemplo de este enfoque.

Note

Si un `BeanTableSchema` no se comporta como espera, active el registro de depuración para `software.amazon.awssdk.enhanced.dynamodb.beans`.

[ImmutableTableSchema](#)

Esta implementación se crea a partir de una clase de datos inmutable. Esto se describe en la sección [???](#) anterior.

Esquema de tabla generado con un generador

Los siguientes `TableSchema` se crean a partir de código mediante un generador. Este enfoque es menos costoso que el enfoque que usa clases de datos anotadas. El enfoque de creación evita el uso de anotaciones y no requiere estándares de JavaBean nomenclatura.

[StaticTableSchema](#)

Esta implementación está creada para clases de datos mutables. En la sección de introducción de esta guía se muestra cómo [crear un StaticTableSchema mediante un generador](#).

[StaticImmutableTableSchema](#)

De forma similar a como se crea un `StaticTableSchema`, se genera una implementación de este tipo de `TableSchema` utilizando un [generador](#) para su uso con clases de datos inmutables.

Esquema de tabla para datos sin un esquema fijo

[DocumentTableSchema](#)

A diferencia de otras implementaciones de `TableSchema`, no define los atributos de una instancia `DocumentTableSchema`. Por lo general, solo especifica las claves principales y los proveedores de convertidores de atributos. Una instancia `EnhancedDocument` proporciona los atributos que se crean a partir de elementos individuales o de una cadena JSON.

Incluir o excluir explícitamente atributos

La API del cliente mejorado de DynamoDB ofrece anotaciones para impedir que los atributos de las clases de datos se conviertan en atributos de una tabla. Con la API, también puede usar un nombre de atributo diferente del nombre de atributo de la clase de datos.

Excluir atributos

Para ignorar los atributos que no se deben asignar a una tabla de DynamoDB, marque el atributo con la anotación `@DynamoDbIgnore`.

```
private String internalKey;

@DynamoDbIgnore
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { this.internalKey = internalKey;}
```

Incluir atributos

Para cambiar el nombre de un atributo utilizado en la tabla de DynamoDB, márkelo con la anotación `@DynamoDbAttribute` y escriba un nombre diferente.

```
private String internalKey;

@DynamoDbAttribute("renamedInternalKey")
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { this.internalKey = internalKey;}
```

Conversión de atributos de control

De forma predeterminada, un esquema de tabla proporciona convertidores para muchos tipos comunes de Java mediante una implementación predeterminada de la interfaz.

[AttributeConverterProvider](#) Puede cambiar el comportamiento predeterminado general con una implementación de `AttributeConverterProvider` personalizada. También puede cambiar el conversor de un solo atributo.

Para obtener una lista de los convertidores disponibles, consulte el documento [AttributeConverter](#) de la interfaz Java.

Proporcionar proveedores de convertidores de atributos personalizados

Puede proporcionar una `AttributeConverterProvider` única o una cadena de `AttributeConverterProvider` ordenadas a través de la anotación `@DynamoDbBean` (`converterProviders = {...}`). Cualquier `AttributeConverterProvider` personalizada debe extender la interfaz `AttributeConverterProvider`.

Tenga en cuenta que si suministra su propia cadena de proveedores de convertidores de atributos, anulará el proveedor de convertidores predeterminado `DefaultAttributeConverterProvider`. Si desea utilizar la funcionalidad del `DefaultAttributeConverterProvider`, debe incluirlo en la cadena.

También es posible anotar el bean con una matriz vacía {}. Esto deshabilita el uso de cualquier proveedor de conversión de atributos, incluido el predeterminado. En este caso, todos los atributos que se van a asignar deben tener su propio convertidor de atributos.

El fragmento siguiente muestra un único proveedor de convertidores.

```
@DynamoDbBean(converterProviders = ConverterProvider1.class)
public class Customer {

}
```

El siguiente fragmento muestra el uso de una cadena de proveedores de convertidores. Como el SDK predeterminado es el último que se proporciona, tiene la prioridad más baja.

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

}
```

Los creadores de esquemas de tablas estáticas tienen un método `attributeConverterProviders()` que funciona de la misma manera. Esto se muestra en el siguiente fragmento.

```
private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            .getter(Customer::getName)
            .setter(Customer::setName))
        .attributeConverterProviders(converterProvider1, converterProvider2)
        .build();
```

Sustituir la asignación de un único atributo

Para sustituir la forma en que se asigna un único atributo, introduzca un `AttributeConverter` para el atributo. Esto anula los convertidores proporcionados por `AttributeConverterProviders` en el esquema de la tabla. Esto añade un conversor

personalizado solo para ese atributo. Otros atributos, incluso los del mismo tipo, no utilizarán ese convertidor salvo que se especifique explícitamente para esos otros atributos.

La anotación `@DynamoDbConvertedBy` se usa para especificar la clase `AttributeConverter` personalizada, como se muestra en el siguiente fragmento.

```
@DynamoDbBean
public class Customer {
    private String name;

    @DynamoDbConvertedBy(CustomAttributeConverter.class)
    public String getName() { return this.name; }
    public void setName(String name) { this.name = name;}
}
```

Los generadores de esquemas estáticos tienen un método `attributeConverter()` de creación de atributos equivalente. Este método toma una instancia de un `AttributeConverter`, como se muestra a continuación.

```
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();
```

Ejemplo

En este ejemplo, se muestra una implementación de `AttributeConverterProvider` que proporciona un conversor de atributos para objetos [java.net.HttpCookie](http://java.net/HttpCookie).

La siguiente clase `SimpleUser` contiene un nombre de atributo `lastUsedCookie` que es una instancia de `HttpCookie`.

El parámetro de las anotaciones de `@DynamoDbBean` muestra las dos clases de `AttributeConverterProvider` que proporcionan convertidores.

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();

```


El `CookieConverterProvider` en el ejemplo siguiente proporciona una instancia de un `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);
    }
}
```

Conversión de códigos

En el método `transformFrom()` de la clase `HttpCookieConverter` siguiente, el código recibe una instancia `HttpCookie` y la transforma en un mapa de DynamoDB que se almacena como un atributo.

El método `transformTo()` recibe un parámetro de mapa de DynamoDB y, a continuación, invoca el constructor `HttpCookie` que requiere un nombre y un 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;
    }
}

```

Cambiar el comportamiento de actualización de los atributos

Puede personalizar el comportamiento de actualización de los atributos individuales al realizar una operación de actualización. [Algunos ejemplos de operaciones de actualización en la API de cliente mejorada de DynamoDB son `updateItem\(\)` y `transactWriteItems\(\)`](#)

Por ejemplo, imagine que desea almacenar en su registro una marca de tiempo de tipo creado en el registro. Sin embargo, desea que su valor se escriba solo si no existe ningún valor para el atributo en la base de datos. En este caso, utiliza el comportamiento de actualización de [WRITE_IF_NOT_EXISTS](#).

En el siguiente ejemplo, se muestra la anotación que añade el comportamiento al `createdOn` atributo.

```

@DynamoDbBean
public class Customer extends GenericRecord {
    private String id;
    private Instant createdOn;

    @DynamoDbPartitionKey

```

```

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

@DynamoDbUpdateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)
public Instant getCreatedOn() { return this.createdOn; }
public void setCreatedOn(Instant createdOn) { this.createdOn = createdOn; }
}

```

Puede declarar el mismo comportamiento de actualización al crear un esquema de tabla estática, como se muestra en el siguiente ejemplo después de la línea de comentario 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();

```

Aplanar atributos de otras clases

Si los atributos de su tabla están repartidos en varias clases Java diferentes, ya sea por herencia o composición, la API del cliente mejorado de DynamoDB proporciona soporte para aplanar los atributos en una sola clase.

Utilizar la herencia

Si sus clases utilizan herencias, utilice los siguientes enfoques para aplanar la jerarquía.

Utilizar objetos bean anotados

Para el método de anotación, ambas clases deben llevar la anotación `@DynamoDbBean` y una de ellas llevar una o más anotaciones de clave principal.

A continuación, se muestran ejemplos de clases de datos que tienen una relación de herencia.

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

La [opción onMethod de Lombok](#) copia anotaciones de DynamoDB basadas en atributos, como `@DynamoDbPartitionKey`, en el código generado.

```
@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 el enfoque de esquema estático, utilice el método `extend()` del generador para contraer los atributos de la clase principal en la clase secundaria. Esto se muestra después de la línea de comentario 1 en el siguiente ejemplo.

```
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::getCreatedAt)

    .setter(org.example.tests.model.inheritance.stat.GenericRecord::setCreatedAt))
    .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();
```

El ejemplo de esquema estático anterior utiliza las siguientes clases de datos. Dado que la asignación se define cuando se construye el esquema estático de la tabla, las clases de datos no requieren anotaciones.

Clases de datos

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;
}
```

Utilizar composiciones

Si sus clases utilizan composiciones, siga los siguientes planteamientos para aplanar la jerarquía.

Utilizar objetos bean anotados

La anotación `@DynamoDbFlatten` aplanar la clase contenida.

Los siguientes ejemplos de clases de datos utilizan la anotación `@DynamoDbFlatten` para añadir de forma eficaz todos los atributos de la clase `GenericRecord` contenida a la clase `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;
}
```

Puede utilizar la anotación aplanar para aplanar tantas clases elegibles diferentes como necesite. Existen las siguientes limitaciones:

- Después de ser aplanados, todos los nombres de atributos deben ser únicos.
- Nunca debe haber más de una clave de partición, clave de clasificación o nombre de tabla.

Usar esquemas estáticos

Cuando construya un esquema de tabla estático, utilice el método `flatten()` del constructor. También debe suministrar los métodos `getter` y `setter` que identifican la clase contenida.

```
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();

```

El ejemplo de esquema estático anterior utiliza las siguientes clases de datos.

Clases de datos

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;
}

```

Puede utilizar el patrón constructor para aplanar tantas clases elegibles diferentes como necesite.

Implicaciones para otro código

Cuando se utiliza el atributo `@DynamoDbFlatten` (o el método constructor `flatten()`), el elemento en DynamoDB contiene un atributo para cada atributo del objeto compuesto. También incluye los atributos del objeto que lo compone.

Por el contrario, si anota una clase de datos con una clase compuesta y no usa `@DynamoDbFlatten`, el elemento se guarda con el objeto compuesto como un atributo único.

Por ejemplo, compare la clase `Customer` que se muestra en el [ejemplo de aplanamiento con composición](#) con y sin aplanamiento del atributo `record`. Puede visualizar la diferencia con JSON según muestra la siguiente tabla.

Con aplanamiento	Con aplanamiento
3 atributos	2 atributos
<pre> { "id": "1", "createdAt": "today", "name": "my name" } </pre>	<pre> { "id": "1", "record": { "createdAt": "today", "name": "my name" } } </pre>

La diferencia es importante si hay otro código que accede a la tabla de DynamoDB y espera encontrar determinados atributos.

Trabajar con atributos anidados

Un atributo anidado de DynamoDB está incrustado en otro atributo. Algunos ejemplos son los elementos de una lista y las entradas de mapa.

En Java, un atributo anidado de DynamoDB corresponde a un miembro de una clase `List` o `Map`. También corresponde a una instancia de tipo complejo, como `Address` o `PhoneNumber`, como se usa en la siguiente clase `Person`.

Clase **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 +  
        '}';  
    }  
}
```

Clase **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 + '\'' +
            '}';
    }
}
```

Clase **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 + '\'' +
        '}';
}
}
```

Atributos anidados de mapa

Usar clases anotadas

Si desea guardar atributos anidados para clases personalizadas, puede anotarlas. La clase `Address` y la clase `PhoneNumber` mostradas anteriormente se anotan solo con la anotación `@DynamoDbBean`. Cuando la API del cliente mejorado de DynamoDB construye el esquema de tabla para la clase `Person` con el siguiente fragmento, la API descubre el uso de las clases `Address` y `PhoneNumber`, y construye las asignaciones correspondientes para trabajar con DynamoDB.

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

Usar esquemas anidados

El enfoque alternativo consiste en utilizar generadores de esquemas de tablas estáticas para cada una de las clases, tal y como se muestra en el código siguiente.

Los esquemas de tabla de las clases `Address` y `PhoneNumber` y son abstractos en el sentido de que no se pueden usar con una tabla de DynamoDB. Esto se debe a que carecen de definiciones

para la clave principal. Sin embargo, se utilizan como esquemas anidados en el esquema de tabla de la clase `Person`.

Tras las líneas de comentario 1 y 2 de la definición de `PERSON_TABLE_SCHEMA`, verá el código que utiliza los esquemas de tabla abstractos. El uso de `documentOf` en el método `EnhanceType.documentOf(...)` no indica que este devuelva un tipo `EnhancedDocument` de la API de documentos mejorados. En este contexto, el método `documentOf(...)` devuelve un objeto que sabe cómo asignar su argumento de clase a y desde los atributos de la tabla de DynamoDB mediante el argumento del esquema de la tabla.

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();

```

Atributos anidados proyectados

Para los métodos `query()` y `scan()`, puede especificar qué atributos desea que se devuelvan en los resultados mediante llamadas a métodos como `addNestedAttributeToProject()` y `attributesToProject()`. La API del cliente mejorada de DynamoDB convierte los parámetros de llamada al método Java [en expresiones de proyección](#) antes de enviar la solicitud.

En el siguiente ejemplo, se rellena la tabla `Person` con dos elementos y, a continuación, se efectúan tres operaciones de análisis.

El primer análisis accede a todos los elementos de la tabla para comparar los resultados con las demás operaciones de análisis.

El segundo análisis aplica el método del constructor [addNestedAttributeToProject\(\)](#) para devolver solo el valor del atributo `street`.

La tercera operación de análisis sigue el método del constructor [attributesToProject\(\)](#) para devolver los datos del atributo de primer nivel, `hobbies`. El tipo de atributo `hobbies` es una lista. Para acceder a cada elemento de la lista, haga una operación `get()` en la 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

Si el método `attributesToProject()` sigue cualquier otro procedimiento de creación que añada los atributos que desee proyectar, la lista de nombres de atributos proporcionada a `attributesToProject()` sustituirá a todos los demás nombres de atributos. Un análisis efectuado con la instancia `ScanEnhancedRequest` del fragmento siguiente devuelve solo datos de hobbies.

```
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]}
```

El siguiente fragmento de código usa primero el método `attributesToProject()`. Este orden preserva todos los demás atributos solicitados.

```
ScanEnhancedRequest attributesPreserved = ScanEnhancedRequest.builder()
```

```

        // Use 'attributesToProject()' first so that the method call does not
        // replace all other attributes
        // that you want to project.
        .attributesToProject("firstName")
        .addNestedAttributeToProject(
            NestedAttributeName.create("addresses", "work", "street"))
        .addAttributeToProject("hobbies")
        .build();
PageIterable<Person> allAttributesResult =
    personDynamoDbTable.scan(attributesPreserved);
allAttributesResult.items().forEach(p ->
    logger.info(p.toString()));

// Logged results.
Person{id=null, firstName='first name 2', lastName='null', age=null,
    addresses={work=Address{street='street 21', city='null', state='null',
    zipCode='null'}}}, phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='first name 1', lastName='null', age=null,
    addresses={work=Address{street='street 11', city='null', state='null',
    zipCode='null'}}}, phoneNumbers=null, hobbies=[hobby 1, hobby 11]}

```

Conserva los objetos vacíos con `@DynamoDbPreserveEmptyObject`

Si guarda un bean en Amazon DynamoDB con objetos vacíos y desea que el SDK vuelva a crear los objetos vacíos al recuperarlos, anote el getter del bean interior con `@DynamoDbPreserveEmptyObject`.

Para ilustrar cómo funciona la anotación, en el ejemplo de código se utilizan los dos beans siguientes.

Beans de ejemplo

La clase de datos siguiente contiene dos campos `InnerBean`. El método de getter, `getInnerBeanWithoutAnno()`, no está anotado con `@DynamoDbPreserveEmptyObject`. El método `getInnerBeanWithAnno()` está 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();
}
}

```

Las instancias de la clase siguiente InnerBean son campos de MyBean y se inicializan como objetos vacíos en el código de ejemplo.

```

@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 + '\'' +
            '}';
    }
}

```

El siguiente ejemplo de código guarda un objeto MyBean con objetos beans internos inicializados en DynamoDB y, a continuación, recupera el elemento. El resultado registrado muestra que el `innerBeanWithoutAnno` no se ha inicializado, sino que se ha creado `innerBeanWithAnno`.

```

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

Puede utilizar la siguiente versión de `StaticTableSchema` de los esquemas de tabla en lugar de las anotaciones de los 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 guardar atributos nulos de los objetos anidados

Puede omitir los atributos nulos de los objetos anidados al guardar un objeto de clase de datos en DynamoDB aplicando la anotación. `@DynamoDbIgnoreNulls` Por el contrario, los atributos de nivel superior con valores nulos nunca se guardan en la base de datos.

Para ilustrar cómo funciona la anotación, en el ejemplo de código se utilizan los dos beans siguientes.

Beans de ejemplo

La clase de datos siguiente contiene dos campos `InnerBean`. El método de getter, `getInnerBeanWithoutAnno()`, no está anotado. El método `getInnerBeanWithIgnoreNullsAnno()` está anotado con `@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();
    }
}
```

Las instancias de la clase `InnerBean` siguiente son campos de `MyBean` y se inicializan como objetos vacíos en el código de ejemplo.

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

El siguiente ejemplo de código crea un objeto `InnerBean` y establece solo uno de sus dos atributos con un 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 los datos de bajo nivel que se envían a DynamoDB, el código registra el mapa de atributos antes de guardar el objeto MyBean.

La salida registrada muestra que `innerBeanWithIgnoreNullsAnno` emite un atributo,

```
innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)})
```

La instancia `innerBeanWithoutAnno` genera dos atributos. Un atributo tiene un valor de 200 y el otro tiene un valor nulo.

```
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
innerBeanFieldString=AttributeValue(NUL=true)})
```

Representación en JSON del mapa de atributos

La siguiente representación en JSON facilita la visualización de los datos guardados en DynamoDB.

```

{
  "id": {
    "S": "1"
  },
  "innerBeanWithIgnoreNullsAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      }
    }
  },
  "innerBeanWithoutAnno": {
    "M": {

```

```

    "innerBeanFieldInteger": {
        "N": "200"
    },
    "innerBeanFieldString": {
        "NULL": true
    }
}
}
}
}

```

Esquema estático alternativo

Puede utilizar la siguiente versión de `StaticTableSchema` de los esquemas de tabla en lugar de las anotaciones de los beans.

```

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();
}
```

Trabaje con documentos JSON con la API de documentos mejorada para DynamoDB

La [API de documentos mejorada](#) para AWS SDK for Java 2.x está diseñada para funcionar con datos orientados a documentos que no tienen un esquema fijo. Sin embargo, también le permite utilizar clases personalizadas para asignar atributos individuales.

La API de documentos mejorada es la sucesora de la [API de documentos](#) de la AWS SDK for Java versión 1.x.

Contenido

- [Introducción al uso de API de documentos mejorada](#)
 - [Crear un DocumentTableSchema y una DynamoDbTable.](#)
- [Crear documentos mejorados](#)
 - [Compilar a partir de una cadena JSON](#)
 - [Construir a partir de elementos individuales](#)
- [Operaciones CRUD](#)
- [Acceder a los atributos mejorados del documento como objetos personalizados](#)
- [Utilizar un EnhancedDocument sin DynamoDB](#)

Introducción al uso de API de documentos mejorada

La API de documentos mejorada requiere las mismas [dependencias](#) que se necesitan para la API de cliente mejorado de DynamoDB. También requiere una [instancia de DynamoDbEnhancedClient](#), como se muestra al principio de este tema.

Como la API de documentos mejorada se publicó con la versión 2.20.3 de AWS SDK for Java 2.x, necesitará esa versión o una superior.

Crear un `DocumentTableSchema` y una `DynamoDbTable`.

Para invocar comandos en una tabla de DynamoDB mediante la API de documentos mejorada, asocie la tabla a un objeto de recurso de [DynamoDBTable<EnhancedDocument>](#) del cliente.

El método `table()` del cliente mejorado crea una instancia `DynamoDbTable<EnhancedDocument>` y requiere parámetros para el nombre de la tabla de DynamoDB y un `DocumentTableSchema`.

El creador de un [DocumentTableSchema](#) requiere una clave de índice principal y uno o más proveedores de convertidores de atributos. El método `AttributeConverterProvider.defaultProvider()` proporciona convertidores para los [tipos predeterminados](#). Debe especificarse incluso si proporciona un proveedor de convertidores de atributos personalizado. Puede añadir una clave de índice secundaria opcional al creador.

El siguiente fragmento de código muestra el código que genera la representación del lado de una tabla `person` de DynamoDB que almacena objetos `EnhancedDocument` sin 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.
```

A continuación, se muestra la representación JSON de un objeto `person` que se utiliza en esta sección.

Objeto `person` JSON

```
{
```

```
"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"
  }
]
}
```

Crear documentos mejorados

Un [EnhancedDocument](#) representa un objeto de tipo documento que tiene una estructura compleja con atributos anidados. Un `EnhancedDocument` requiere atributos de nivel superior que coincidan con los atributos clave principales especificados para el `DocumentTableSchema`. El contenido restante es arbitrario y puede consistir en atributos de nivel superior y también en atributos profundamente anidados.

Para crear una instancia `EnhancedDocument`, utilice un creador que proporciona varias formas de añadir elementos.

Compilar a partir de una cadena JSON

Con una cadena JSON, puede construir un `EnhancedDocument` en una llamada al método. El siguiente fragmento de código crea un `EnhancedDocument` a partir de una cadena JSON devuelta por el método auxiliar `jsonPerson()`. El método `jsonPerson()` devuelve la versión de cadena JSON del [objeto persona](#) mostrado anteriormente.

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

Construir a partir de elementos individuales

Alternativamente, puede construir una instancia `EnhancedDocument` a partir de componentes individuales utilizando métodos seguros de tipo del creador.

En el ejemplo siguiente, se crea un documento mejorado `person` similar al que se crea a partir de la cadena JSON del ejemplo 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"));
}

```

```
}
```

Operaciones CRUD

Una vez definida una instancia `EnhancedDocument`, puede guardarla en una tabla de DynamoDB. El siguiente fragmento de código utiliza el [PersonDocument](#) que se creó a partir de elementos individuales.

```
documentDynamoDbTable.putItem(personDocument);
```

Después de leer una instancia de documento mejorada de DynamoDB, puede extraer los valores individuales de los atributos utilizando getters como se muestra en el siguiente fragmento de código que accede a los datos guardados del `personDocument`. Como alternativa, puede extraer el contenido completo en una cadena JSON, como se muestra en la última parte del código de ejemplo.

```
// 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"}]}

```

Las instancias de `EnhancedDocument` se pueden usar con cualquier método [DynamoDbTable](#) o [DynamoDBEnhancedClient](#) en lugar de clases de datos mapeadas.

Acceder a los atributos mejorados del documento como objetos personalizados

Además de proporcionar una API para leer y escribir atributos con estructuras sin esquema, la API de documentos mejorada te permite convertir atributos desde y hacia instancias de clases personalizadas.

La API del cliente mejorado utiliza `AttributeConverterProviders` y `AttributeConverters` que se mostraron en la sección de [conversión de atributos de control](#) como parte de la API del cliente mejorado de DynamoDB.

En el ejemplo siguiente, utilizamos a `CustomAttributeConverterProvider` con su clase `AddressConverter` anidada para convertir objetos `Address`.

En este ejemplo se muestra que se pueden mezclar datos de clases y también datos de estructuras que se crean según sea necesario. En este ejemplo también se muestra que las clases personalizadas se pueden utilizar en cualquier nivel de una estructura anidada. Los objetos Address de este ejemplo son valores que se utilizan en un 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'}

```

CustomAttributeConverterProvider code

```

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;
    }
}
}
```

Clase **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 proporciona direcciones

El siguiente método auxiliar proporciona el mapa que utiliza instancias `Address` personalizadas para los valores en lugar de instancias `Map<String, String>` genéricas para los 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);
}
```

Utilizar un **EnhancedDocument** sin DynamoDB

Aunque normalmente se utiliza una instancia de un `EnhancedDocument` para leer y escribir elementos de DynamoDB de tipo documento, también se puede utilizar con independencia de DynamoDB.

Puede utilizar `EnhancedDocuments` por su capacidad de convertir cadenas JSON u objetos personalizados en mapas de `AttributeValues` de bajo nivel, como se muestra en el siguiente ejemplo.

```
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

Cuando utilice un documento mejorado independiente de una tabla de DynamoDB, asegúrese de establecer explícitamente los proveedores de convertidores de atributos en el creador.

Por el contrario, el esquema de la tabla de documentos proporciona los proveedores de conversión cuando se utiliza un documento mejorado con una tabla de DynamoDB.

Usar extensiones

La API del cliente mejorado de DynamoDB admite extensiones de complementos que proporcionan funciones que van más allá de las operaciones de mapeo. Las extensiones tienen dos métodos de enlace, `beforeWrite()` y `afterRead()`. `beforeWrite()` modifica una operación de escritura antes de que se produzca y el método `afterRead()` modifica los resultados de una operación de lectura después de que se produzca. Como algunas operaciones (como las actualizaciones de elementos) hacen una escritura y una lectura, se recurre a ambos métodos de enlace.

Las extensiones se cargan en el orden en que se especificaron en el creador de clientes mejorado. El orden de carga puede ser importante porque una extensión puede actuar sobre valores que han sido transformados por una extensión anterior.

La API de cliente mejorada incluye un conjunto de extensiones de complementos que se encuentran en el paquete [extensions](#). De forma predeterminada, el cliente mejorado carga [VersionedRecordExtension](#) y [AtomicCounterExtension](#). Puede anular el comportamiento predeterminado con el creador de clientes mejorado y cargar cualquier extensión. También puede no especificar ninguna si no desea utilizar las extensiones predeterminadas.

Si carga sus propias extensiones, el cliente mejorado no carga ninguna extensión predeterminada. Si desea el comportamiento proporcionado por alguna de las extensiones predeterminadas, deberá añadirla explícitamente a la lista de extensiones.

En el siguiente ejemplo, una extensión personalizada denominada `verifyChecksumExtension` se carga después de la `VersionedRecordExtension`, que normalmente se carga de forma predeterminada por sí misma. La `AtomicCounterExtension` no se carga en este ejemplo.

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

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

VersionedRecordExtension

La `VersionedRecordExtension` se carga de forma predeterminada e incrementará y rastreará el número de versión de un elemento a medida que los elementos se escriban en la base de datos. Se añadirá una condición a cada escritura que hará que esta falle si el número de versión del elemento persistido real no coincide con el valor que la aplicación leyó por última vez. Este comportamiento proporciona efectivamente un bloqueo positivo para las actualizaciones de elementos. Si otro proceso actualiza un elemento entre el momento en que el primer proceso ha leído el elemento y está escribiendo una actualización en él, la escritura fallará.

Para especificar qué atributo usar para rastrear el número de versión del elemento, etiquete un atributo numérico en el esquema de la tabla.

El siguiente fragmento especifica que el atributo `version` debe contener el número de versión del artículo.

```
@DynamoDbVersionAttribute
public Integer getVersion() {...};
public void setVersion(Integer version) {...};
```

El enfoque equivalente del esquema de tabla estático se muestra en el siguiente fragmento.

```
.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

La `AtomicCounterExtension` se carga de forma predeterminada e incrementa un atributo numérico etiquetado cada vez que se escribe un registro en la base de datos. Se pueden especificar los valores de inicio y de incremento. Si no se especifica ningún valor, el valor inicial se establece en 0 y el valor del atributo se incrementa en 1.

Para especificar qué atributo es un contador, etiquete un atributo de tipo `Long` en el esquema de la tabla.

En el siguiente fragmento se muestra el uso de los valores de inicio e incremento predeterminados para el atributo `counter`.

```
@DynamoDbAtomicCounter
public Long getCounter() {...};
public void setCounter(Long counter) {...};
```

El enfoque del esquema de tabla estático se muestra en el siguiente fragmento. La extensión del contador atómico utiliza un valor inicial de 10 e incrementa el valor en 5 cada vez que se escribe el registro.

```
.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

La `AutoGeneratedTimestampRecordExtension` actualiza automáticamente los atributos de tipo [Instant](#) con una marca de tiempo actual cada vez que el elemento se escribe correctamente en la base de datos.

Esta extensión no se carga de forma predeterminada. Por lo tanto, debe especificarla como una extensión personalizada al crear el cliente mejorado, como se muestra en el primer ejemplo de este tema.

Para especificar qué atributo actualizar con la marca de tiempo actual, etiquete el atributo `Instant` en el esquema de la tabla.

El atributo `lastUpdate` es el objetivo del comportamiento de las extensiones en el siguiente fragmento. Tenga en cuenta el requisito de que el atributo debe ser de tipo `Instant`.

```
@DynamoDbAutoGeneratedTimestampAttribute
public Instant getLastUpdate() {...}
public void setLastUpdate(Instant lastUpdate) {...}
```

El enfoque equivalente del esquema de tabla estático se muestra en el siguiente fragmento.

```
.addAttribute(Instant.class, a -> a.name("lastUpdate")
                .getter(Customer::getLastUpdate)
                .setter(Customer::setLastUpdate)
```

```

// Applying the 'autoGeneratedTimestamp' tag to
the attribute.

.tags(AutoGeneratedTimestampRecordExtension.AttributeTags.autoGeneratedTimestampAttribute())

```

Extensiones personalizadas

La siguiente clase de extensión personalizada muestra un método `beforeWrite()` que usa una expresión de actualización. Después de la línea de comentarios 2, creamos un `SetAction` para establecer el atributo `registrationDate` si el elemento de la base de datos aún no tiene un atributo `registrationDate`. Siempre que se actualiza un objeto `Customer`, la extensión se asegura de que se establece una `registrationDate`.

```

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();
    }
}

```

Utilizar la API de cliente mejorado de DynamoDB de forma asíncrona

Si la aplicación requiere llamadas asíncronas y sin bloqueo a DynamoDB, puede usar [DynamoDBEnhancedAsyncClient](#). Es similar a la implementación síncrona, pero con las siguientes diferencias clave:

1. Al compilar `DynamoDbEnhancedAsyncClient`, debe proporcionar la versión asíncrona del cliente estándar, `DynamoDbAsyncClient`, como se muestra en el siguiente fragmento.

```

DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient.builder()
        .dynamoDbClient(dynamoDbAsyncClient)
        .build();

```

2. Los métodos que devuelven un único objeto de datos devuelven un `CompletableFuture` del resultado en lugar de solo el resultado. De este modo, la aplicación podrá hacer otras tareas sin bloquear el resultado. En el siguiente fragmento se muestra el método `getItem()` asíncrono.

```

CompletableFuture<Customer> result = customerDynamoDbTable.getItem(customer);
// Perform other work here.
return result.join(); // Now block and wait for the result.

```

3. Los métodos que devuelven listas paginadas de resultados devuelven un [SdkPublisher](#), en lugar de un [SdkIterable](#) que el método síncrono `DynamoDbEnhancedClient` devuelve para los mismos métodos. A continuación, su aplicación puede suscribir un controlador a ese publicador para tratar los resultados de forma asíncrona sin tener que bloquearse.

```

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 ver un ejemplo más completo de cómo trabajar con el `SdkPublisher` API, consulte el [ejemplo](#) de la sección que trata sobre el método `scan()` asíncrono de esta guía.

Anotaciones de clases de datos

En la tabla siguiente se enumeran las anotaciones que se pueden usar en las clases de datos y se proporcionan enlaces a información y ejemplos en esta guía. La tabla está ordenada alfabéticamente en orden ascendente por nombre de anotación.

Anotaciones de clases de datos utilizadas en esta guía

Nombre de la anotación	La anotación se aplica a 1	¿Qué hace?	Dónde aparece en esta guía
<code>DynamoDbAtomicCounter</code>	atributo 2	Incrementa un atributo numérico etiquetado cada vez que se escribe un registro en la base de datos.	Introducción y discusión.
<code>DynamoDbAttribute</code>	atributo	Define o cambia el nombre de una propiedad de bean que está asignada a un atributo de tabla de DynamoDB.	<ul style="list-style-type: none"> • Discusión inicial. • Sección de introducción: consulte la nota. • Ejemplos del método In Query en MovieActor clase.
<code>DynamoDbAutoGeneratedTimestampAttribute</code>	atributo	Actualiza un atributo etiquetado con una marca de tiempo actual cada vez que el elemento se escribe correctamente en la base de datos.	Introducción y discusión.

Nombre de la anotación	La anotación se aplica a 1	¿Qué hace?	Dónde aparece en esta guía
DynamoDbBean	class	Marca una clase de datos como asignable a un esquema de tabla.	Primer uso en la clase Customer en la sección Comenzar. A lo largo de la guía aparecen varios usos.
DynamoDbConvertedBy	atributo	Asocia un <code>AttributeConverter</code> personalizado al atributo anotado.	Discusión inicial y ejemplo.
DynamoDbFlatten	atributo	Aplana todos los atributos de una clase de datos de DynamoDB independiente y los agrega como atributos de nivel superior al registro que se lee y escribe en la base de datos.	<ul style="list-style-type: none"> • Discusión inicial. • Implicaciones para otro código.
DynamoDbIgnore	atributo	Hace que el atributo quede sin asignar.	<ul style="list-style-type: none"> • Discusión inicial. • Úselo en la ProductCatalog clase.
DynamoDbIgnoreNulls	atributo	Impide guardar los atributos nulos de los DynamoDb objetos anidados.	Discusión y ejemplos.

Nombre de la anotación	La anotación se aplica a 1	¿Qué hace?	Dónde aparece en esta guía
DynamoDbImmutable	class	Marca una clase de datos inmutable como asignable a un esquema de tabla.	<ul style="list-style-type: none"> • Introducción a la anotación. • Úselo en la ProductCatalog clase. • Uso con Lombok.
DynamoDbPartitionKey	atributo	Marca un atributo como clave de partición principal (clave hash) de la DynamoDb tabla.	<ul style="list-style-type: none"> • Primer uso en la clase Customer en la sección Comenzar. • Con Lombok.
DynamoDbP reserveEmptyObject	atributo	Especifica que, si no hay datos presentes para el objeto asignado al atributo anotado, el objeto debe inicializarse con todos los campos nulos.	Discusión y ejemplos.
DynamoDbS econdaryPartitionKey	atributo	Marca un atributo como clave de partición para un índice secundario global.	<ul style="list-style-type: none"> • Uso en índices secundarios y ejemplo. • En ejemplos del método Query. • En ejemplo de Lombok • Con clases inmutables.

Nombre de la anotación	La anotación se aplica a 1	¿Qué hace?	Dónde aparece en esta guía
DynamoDbSecondarySortKey	atributo	Marca un atributo como clave de clasificación opcional para un índice secundario global o local.	<ul style="list-style-type: none"> • Uso en índices secundarios y ejemplo. • En ejemplos del método Query. • En ejemplo de Lombok. • Con clases inmutables.
DynamoDbSortKey	atributo	Marca un atributo como clave de clasificación principal opcional (clave de rango).	<ul style="list-style-type: none"> • Sección de introducción sobre la clase Customer. • Con clases inmutables. • En ejemplo de Lombok. • En ejemplos del método Query.
DynamoDbUpdateBehavior	atributo	Especifica el comportamiento cuando este atributo se actualiza como parte de una operación de «actualización», por ejemplo UpdateItem.	Introducción y ejemplo.
DynamoDbVersionAttribute	atributo	Incrementa el número de versión de un artículo.	Introducción y discusión.

¹ Puede aplicar anotaciones a nivel de atributo al captador o al definidor, pero no a ambos. Esta guía muestra las anotaciones en los captadores.

² El término `property` se utiliza normalmente para un valor encapsulado en una JavaBean clase de datos. Sin embargo, en esta guía se utiliza el término `attribute` en su lugar para mantener la coherencia con la terminología utilizada por DynamoDB.

Trabaja con Amazon EC2

En esta sección se proporcionan ejemplos de programación [Amazon EC2](#) que utilizan la versión AWS SDK for Java 2.x.

Temas

- [Administrar instancias Amazon EC2](#)
- [Zonas de uso Regiones de AWS y disponibilidad](#)
- [Trabaje con grupos de seguridad en Amazon EC2](#)
- [Trabajar con metadatos de la instancia de Amazon EC2](#)

Administrar instancias Amazon EC2

Crear una instancia

Cree una nueva instancia Amazon EC2 llamando al método [runInstances](#) de [Ec2Client](#), proporcionando un objeto [RunInstancesRequest](#) que contenga la [Imagen de máquina de Amazon \(AMI\)](#) que se va a usar y un [tipo de instancia](#).

Importaciones

```
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 "";
}
```

Consulte el [ejemplo completo](#) en GitHub.

Iniciar una instancia

Para iniciar una instancia Amazon EC2, llame al método [startInstances](#) de `Ec2Client`, proporcionando un objeto [StartInstancesRequest](#) que contenga el ID de la instancia que se va a iniciar.

Importaciones

```
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);
}
```

Consulte el [ejemplo completo](#) en GitHub.

Detener una instancia

Para detener una instancia Amazon EC2, llame al método [stopInstances](#) de `Ec2Client`, proporcionando un objeto [StopInstancesRequest](#) que contenga el ID de la instancia que se va a detener.

Importaciones

```
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);
}
```

Consulte el [ejemplo completo](#) en GitHub.

Reiniciar una instancia

Para reiniciar una instancia Amazon EC2, llame al método [de rebootInstances](#), proporcionando un objeto [RebootInstancesRequest](#) que contenga el ID de la instancia que se va a reiniciar.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Describir instancias

Para mostrar sus instancias, cree un objeto [DescribeInstancesRequest](#) y llame al método [describeInstances](#) de `Ec2Client`. Se devolverá un objeto [DescribeInstancesResponse](#) que puede utilizar para mostrar las instancias Amazon EC2 de su cuenta y región.

Las instancias se agrupan por reserva. Cada reserva se corresponde con la llamada a `startInstances` que lanzó la instancia. Para mostrar sus instancias, primero debe llamar al método `reservations` de la clase `DescribeInstancesResponse` y después llamar a `instances` en cada objeto [Reservation](#) devuelto.

Importaciones

```
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());
                }
            }
        } while (response.nextToken() != null);
    } catch (Ec2Exception e) {
        e.printStackTrace();
    }
}
```

```
        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);
}
}
```

Los resultados se paganinan; puede obtener más resultados pasando el valor devuelto del método `nextToken` del objeto resultante al método `nextToken` del nuevo objeto de la solicitud, usando el nuevo objeto de la solicitud en la siguiente llamada a `describeInstances`.

Consulte el [ejemplo completo](#) en GitHub.

Monitorear una instancia

Puede monitorizar distintos aspectos de las instancias Amazon EC2, como el uso de la CPU y la red, la memoria disponible y el espacio en disco restante. Para obtener más información sobre la supervisión de instancias, consulte [Supervisión Amazon EC2](#) en la Guía del usuario de Amazon EC2 para instancias Linux.

Para iniciar la monitorización de una instancia, debe crear un objeto [MonitorInstancesRequest](#) con el ID de la instancia que se va a monitorizar y pasarlo al método [monitorInstances](#) de `Ec2Client`.

Importaciones

```
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);
}
```

Consulte el [ejemplo completo](#) en GitHub.

Detener la monitorización de instancias

Para detener la monitorización de una instancia, cree un objeto [UnmonitorInstancesRequest](#) con el ID de la instancia cuya monitorización se va a detener y pase el objeto al método [unmonitorInstances](#) de `Ec2Client`.

Importaciones

```
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);
}
```

Consulte el [ejemplo completo](#) en GitHub.

Más información

- [RunInstances](#) en la referencia de la API de Amazon EC2
- [DescribeInstances](#) en la referencia de la API de Amazon EC2
- [StartInstances](#) en la referencia de la API de Amazon EC2
- [StopInstances](#) en la referencia de la API de Amazon EC2
- [RebootInstances](#) en la referencia de la API de Amazon EC2
- [MonitorInstances](#) en la referencia de la API de Amazon EC2
- [UnmonitorInstances](#) en la referencia de la API de Amazon EC2

Zonas de uso Regiones de AWS y disponibilidad

Describir regiones

Para mostrar las regiones disponibles para su cuenta, llame al método `describeRegions` de `Ec2Client`. Este método devuelve un objeto [DescribeRegionsResponse](#). Llame al método `regions` del objeto devuelto para obtener una lista de objetos [Region](#) que representan cada región.

Importaciones

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

Consulte el [ejemplo completo](#) en GitHub

Describir zonas de disponibilidad

Para mostrar las zonas de disponibilidad disponibles para su cuenta, llame al método `describeAvailabilityZones` de `Ec2Client`. Este método devuelve un objeto [DescribeAvailabilityZonesResponse](#). Llame a su `availabilityZones` método para obtener una lista de [AvailabilityZone](#) objetos que representan cada zona de disponibilidad.

Importaciones

```
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

Crear el `Ec2Client`.

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

A continuación, llame a `describeAvailabilityZones ()` y recupere los 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();
});
```

Consulte el [ejemplo completo](#) en GitHub.

Describir cuentas

Para obtener una lista de la información relacionada con EC2 sobre su cuenta, llame al método `describeAccountAttributes` del `Ec2Client`. Este método devuelve un [DescribeAccountAttributesResponse](#) objeto. Invoca este `accountAttributes` método de objetos para obtener una lista de [AccountAttribute](#) objetos. Puede recorrer la lista en iteraciones para recuperar un `AccountAttribute` objeto.

Puede obtener los valores de los atributos de su cuenta invocando el método del `AccountAttribute` `attributeValues` objeto. Este método devuelve una lista de [AccountAttributeValue](#) objetos. Puede recorrer en iteración esta segunda lista para mostrar el valor de los atributos (consulte el siguiente ejemplo de código).

Importaciones

```
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.print("\n The name of the attribute is " +
attribute.attributeName());
            attribute.attributeValues().forEach(
                myValue -> System.out.print("\n The value of the attribute is "
+ myValue.attributeValue()));
        });

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Más información

- [Regiones y zonas de disponibilidad](#) en la guía Amazon EC2 del usuario de instancias de Linux
- [DescribeRegions](#) en la referencia Amazon EC2 de la API
- [DescribeAvailabilityZones](#) en la referencia Amazon EC2 de la API

Trabaje con grupos de seguridad en Amazon EC2

Creación de un grupo de seguridad

Para crear un grupo de seguridad, llame al `createSecurityGroup` método de `Ec2Client` con un [CreateSecurityGroupRequest](#) que contenga el nombre de la clave.

Importaciones

```
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);
```

Consulte el [ejemplo completo](#) en GitHub

Configurar un grupo de seguridad

Un grupo de seguridad puede controlar el tráfico entrante (entrada) y saliente (salida) de sus instancias. Amazon EC2

Para agregar reglas de entrada a su grupo de seguridad, utilice el `authorizeSecurityGroupIngress` método de `Ec2Client`, proporcionando el nombre del grupo de seguridad y las reglas de acceso ([IpPermission](#)) que desee asignarle dentro de un objeto. [AuthorizeSecurityGroupIngressRequest](#) El siguiente ejemplo muestra cómo añadir permisos de IP a un grupo de seguridad.

Importaciones

```
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

Primero, cree un `Ec2Client`

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

Luego, use el método `authorizeSecurityGroupIngress` de `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 añadir una regla de salida al grupo de seguridad, proporcione datos similares en uno y otro método del [AuthorizeSecurityGroupEgressRequest](#) `Ec2Client`. `authorizeSecurityGroupEgress`

Consulte el ejemplo [completo](#) en GitHub

Describir grupos de seguridad

Para describir los grupos de seguridad o para obtener información sobre ellos, llame al método `describeSecurityGroups` de `Ec2Client`. Devuelve un [DescribeSecurityGroupsResponse](#) que puede utilizar para acceder a la lista de grupos de seguridad llamando a su `securityGroups` método, que devuelve una lista de [SecurityGroup](#) objetos.

Importaciones

```
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);
}
}
```

Consulte el [ejemplo completo](#) en GitHub.

Eliminación de un grupo de seguridad

Para eliminar un grupo de seguridad, llame al `deleteSecurityGroup` método `Ec2Client` y pásele un [DeleteSecurityGroupRequest](#) que contenga el ID del grupo de seguridad que se va a eliminar.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub

Más información

- [Amazon EC2 Grupos de seguridad](#) en la guía Amazon EC2 del usuario de instancias de Linux
- [Autorice el tráfico entrante para sus instancias de Linux](#) en la Guía del Amazon EC2 usuario de instancias de Linux
- [CreateSecurityGroup](#) en la referencia de la Amazon EC2 API
- [DescribeSecurityGroups](#) en la referencia Amazon EC2 de la API
- [DeleteSecurityGroup](#) en la referencia Amazon EC2 de la API
- [AuthorizeSecurityGroupIngress](#) en la referencia Amazon EC2 de la API

Trabajar con metadatos de la instancia de Amazon EC2

Un cliente Java SDK para el servicio de metadatos de la instancia de Amazon EC2 (cliente de metadatos) permite a sus aplicaciones acceder a los metadatos de su instancia EC2 local. El cliente de metadatos trabaja con la instancia local de [IMDSv2](#) (Instance Metadata Service v2) y utiliza peticiones orientadas a la sesión.

Hay dos clases de clientes disponibles en el SDK. El síncrono [Ec2MetadataClient](#) es para operaciones de bloqueo, y el [Ec2MetadataAsyncClient](#) es para casos prácticos asíncronos, no de bloqueo.

Introducción

Para usar el cliente de metadatos, agregue el artefacto Maven `imds` a su proyecto. También necesita clases para una [SdkHttpClient](#) (o una [SdkAsyncHttpClient](#) para la variante asíncrona) de la ruta de clases.

El siguiente XML de Maven muestra fragmentos de dependencia para utilizar el [URLConnectionHttpClient](#) síncrono junto con la dependencia para clientes de metadatos.

```
<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>
```

Busca en el [repositorio central de Maven](#) la última versión del artefacto bom.

Para usar un cliente HTTP asíncrono, sustituya el fragmento de dependencia del artefacto `url-connection-client`. Por ejemplo, el siguiente fragmento de código incluye la implementación [NettyNioAsyncHttpClient](#).

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

Utilizar el cliente de metadatos

Crear una instancia de un cliente de metadatos

Puede crear una instancia de un sistema síncrono `Ec2MetadataClient` cuando solo haya una implementación de la interfaz `SdkHttpClient` en la ruta de clases. Para ello, llame al método estático `Ec2MetadataClient#create()`, como se muestra en el siguiente fragmento de código.

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

Si su aplicación tiene varias implementaciones de la interfaz `SdkHttpClient` o `SdkHttpAsyncClient`, debe especificar una implementación para que la utilice el cliente de metadatos, como se muestra en la sección [the section called “Cliente HTTP configurable”](#).

Note

Para la mayoría de los clientes de servicios, como Amazon S3, el SDK para Java añade automáticamente las implementaciones de la interfaz `SdkHttpClient` o `SdkHttpAsyncClient`. Si su cliente de metadatos usa la misma implementación, `Ec2MetadataClient#create()` funcionará. Si necesita una implementación diferente, debe especificarla al crear el cliente de metadatos.

Enviar solicitudes

Para recuperar metadatos de la instancia, instancie la clase `EC2MetadataClient` y llame al método `get` con un parámetro de ruta que especifique la [categoría de metadatos de la instancia](#).

En el siguiente ejemplo, se imprime el valor asociado a la clave `ami-id` de la consola.

```
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.
```

Si la ruta no es válida, el método `get` genera una excepción.

Puede reutilizar la misma instancia de cliente para varias solicitudes, pero ejecute `close` en el cliente cuando ya no sea necesario, con el fin de liberar recursos. Después de llamar al método `close`, la instancia de cliente ya no se puede usar.

Analizar las respuestas

Los metadatos de la instancia EC2 se pueden generar en diferentes formatos. Los más utilizados son el texto sin formato y JSON. Los clientes de metadatos ofrecen formas de trabajar con esos formatos.

Como se muestra en el ejemplo siguiente, utilice el método `asString` para obtener los datos como una cadena Java. También puede usar el método `asList` para separar una respuesta de texto sin formato que devuelva varias líneas.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/");
String fullResponse = response.asString();
List<String> splits = response.asList();
```

Si la respuesta está en JSON, use el método `Ec2MetadataResponse#asDocument` para analizarla y convertirla en una instancia de [Document](#), como se muestra en el fragmento de código siguiente.

```
Document fullResponse = response.asDocument();
```

Si el formato de los metadatos no está en JSON, se producirá una excepción. Si la respuesta se ha analizado correctamente, puede utilizar la [API de documentos](#) para inspeccionarla con más detalle. Consulte la [tabla de categorías de metadatos](#) de instancia para saber qué categorías de metadatos ofrecen respuestas con formato JSON.

Configurar un cliente de metadatos

Reintentos

Puede configurar un cliente de metadatos con un mecanismo de reintento. Si lo hace, el cliente podrá reintentar automáticamente las solicitudes que no puedan completarse por motivos inesperados. De forma predeterminada, el cliente lo vuelve a intentar tres veces en caso de una solicitud fallida, con un tiempo de retroceso exponencial entre intentos.

Si su caso de uso requiere un mecanismo de reintento diferente, puede personalizar el cliente mediante el método `retryPolicy` de su generador. Por ejemplo, el siguiente ejemplo muestra un cliente síncrono configurado con un retraso fijo de dos segundos entre intentos y cinco reintentos.

```
BackoffStrategy fixedBackoffStrategy =
    FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(retryPolicyBuilder ->
            retryPolicyBuilder.numRetries(5)
                .backoffStrategy(fixedBackoffStrategy))
```

```
.build();
```

Hay varias [BackoffStrategies](#) que puede utilizar con un cliente de metadatos.

También puede inhabilitar por completo el mecanismo de reintento, como se muestra en el siguiente fragmento de código.

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .retryPolicy(Ec2MetadataRetryPolicy.none())  
        .build();
```

El uso de `Ec2MetadataRetryPolicy#none()` desactiva la política de reintentos predeterminada para que el cliente de metadatos no efectúe reintentos.

Versión de IP

De forma predeterminada, un cliente de metadatos usa el punto de conexión IPV4 en `http://169.254.169.254`. Para que el cliente utilice la versión IPV6, utilice el método `endpointMode` o `endpoint` del constructor. Si se llaman ambos métodos en el constructor se produce una excepción.

Los ejemplos siguientes muestran ambas opciones de IPV6.

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .endpointMode(EndpointMode.IPV6)  
        .build();
```

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .endpoint(URI.create("http://[fd00:ec2::254]"))  
        .build();
```

Características principales

Cliente asíncrono

Para utilizar la versión no bloqueante del cliente, instancie una instancia de la clase `Ec2MetadataAsyncClient`. El código del siguiente ejemplo crea un cliente asíncrono con la configuración predeterminada y utiliza el método `get` para recuperar el valor de la clave `ami-id`.

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

El valor de `java.util.concurrent.CompletableFuture` devuelto por el método `get` se completa cuando se devuelve la respuesta. En el ejemplo siguiente, se imprimen los metadatos `ami-id` en la consola.

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

Cliente HTTP configurable

El generador de cada cliente de metadatos tiene un método `httpClient` que puede utilizar para proporcionar un cliente HTTP personalizado.

El ejemplo siguiente muestra el código para una instancia `URLConnectionHttpClient` personalizada.

```
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.
```

En el ejemplo siguiente, se muestra el código de una instancia `NettyNioAsyncHttpClient` personalizada con un cliente de metadatos asíncrono.

```
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.
```

El tema [the section called “Clientes de HTTP”](#) de esta guía proporciona detalles sobre cómo configurar los clientes de HTTP disponibles en el SDK para Java.

Almacenamiento en caché de tokens

Como los clientes de metadatos utilizan IMDSv2, todas las solicitudes están asociadas a una sesión. Una sesión se define mediante un token que tiene una caducidad, que el cliente de metadatos administra en su lugar. Cada solicitud de metadatos reutiliza automáticamente el token hasta que caduque.

De forma predeterminada, un token dura seis horas (21 600 segundos). Le recomendamos que mantenga el valor predeterminado de tiempo de vida, a menos que su caso práctico específico requiera una configuración avanzada.

Si es necesario, configure la duración utilizando el método de generador `tokenTtl`. Por ejemplo, el código del siguiente fragmento crea un cliente con una duración de sesión de cinco minutos.

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

Si omite llamar al método `tokenTtl` en el generador, se utilizará en su lugar la duración predeterminada de 21 600.

Trabajar con IAM

En esta sección se proporcionan ejemplos de programación de AWS Identity and Access Management (IAM) mediante AWS SDK for Java 2.x.

AWS Identity and Access Management (IAM) permite controlar de forma segura el acceso de sus usuarios a servicios y recursos de AWS. Con IAM puede crear y administrar usuarios y grupos de AWS, así como utilizar permisos para conceder o denegar el acceso de estos a los recursos de AWS. Para obtener instrucciones completas de IAM, consulte la [Guía del usuario de IAM](#).

Los siguientes ejemplos incluyen únicamente el código necesario para demostrar cada técnica. El [código de ejemplo completo está disponible en GitHub](#). Desde allí, puede descargar un único archivo de código fuente o clonar el repositorio localmente para obtener todos los ejemplos para compilarlos y ejecutarlos.

Temas

- [Administrar las claves de IAM acceso](#)
- [Administrar usuarios de IAM](#)
- [Cree políticas de IAM con AWS SDK for Java 2.x](#)
- [Trabajar con políticas de IAM](#)
- [Trabajar con certificados IAM de servidor](#)

Administrar las claves de IAM acceso

Creación de una clave de acceso

Para crear una clave de IAM acceso, llama al `IamClient`'s `createAccessKey` método con un [CreateAccessKeyRequest](#) objeto.

Note

Debe establecer la región en `AWS_GLOBAL` para que las `IamClient` llamadas funcionen, ya que IAM es un servicio global.

Importaciones

```
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 "";
```

Consulte el [ejemplo completo](#) en GitHub

Mostrar claves de acceso

Para enumerar las claves de acceso de un usuario determinado, cree un [ListAccessKeysRequest](#) objeto que contenga el nombre de usuario del que desee enumerar las claves y páselo al `IamClient`'s `listAccessKeys` método.

Note

Si no le proporciona un nombre de usuario `listAccessKeys`, intentará enumerar las claves de acceso asociadas a la persona Cuenta de AWS que firmó la solicitud.

Importaciones

```
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);
}
```

Los resultados de `listAccessKeys` están paginados (con un máximo de 100 registros por llamada). Puede llamar al `isTruncated` [ListAccessKeysResponse](#) objeto devuelto para comprobar si la consulta arrojó menos resultados de los disponibles. En tal caso, llame a `marker` en el objeto `ListAccessKeysResponse` y úselo al crear una nueva solicitud. Utilice esta nueva solicitud en la siguiente invocación de `listAccessKeys`.

Consulta el [ejemplo completo](#) en GitHub.

Recuperar el momento en que se usó por última vez una clave de acceso

Para saber la hora en que se utilizó una clave de acceso por última vez, llame al `IamClient`'s `getAccessKeyLastUsed` método con el identificador de la clave de acceso (que se puede transferir mediante un [GetAccessKeyLastUsedRequest](#) objeto).

A continuación, puedes usar el [GetAccessKeyLastUsedResponse](#) objeto devuelto para recuperar la última vez que se usó la clave.

Importaciones

```
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");
}
```

Consulta el [ejemplo completo](#) en GitHub.

Activar o desactivar claves de acceso

Puede activar o desactivar una clave de acceso creando un [UpdateAccessKeyRequest](#) objeto, proporcionando el ID de la clave de acceso y, si lo desea, el nombre de usuario y el que desee y [status](#), a continuación, pasando el objeto de solicitud al `IamClient`'s `updateAccessKey` método.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub

Eliminación de una clave de acceso

Para eliminar permanentemente una clave de acceso, llame al `IamClient`'s `deleteKey` método y suministre una [DeleteAccessKeyRequest](#) que contenga el ID y el nombre de usuario de la clave de acceso.

Note

Una vez eliminada una clave, ya no se puede recuperar ni utilizar. Para desactivar temporalmente una clave para poder volver a activarla más adelante, utilice el [updateAccessKey](#) método en su lugar.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub

Más información

- [CreateAccessKey](#) en la referencia IAM de la API
- [ListAccessKeys](#) en la referencia IAM de la API
- [GetAccessKeyLastUsed](#) en la referencia IAM de la API
- [UpdateAccessKey](#) en la referencia IAM de la API
- [DeleteAccessKey](#) en la referencia IAM de la API

Administrar usuarios de IAM

Crear un usuario

Cree un nuevo IAM usuario proporcionando el nombre de usuario al `IamClient` `createUser` método mediante un [CreateUserRequest](#) objeto que contenga el nombre de usuario.

Importaciones

```
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 "";
}
```

Consulte el [ejemplo completo](#) en GitHub.

Listar usuarios de

Para enumerar los IAM usuarios de tu cuenta, crea una nueva [ListUsersRequest](#) y pásala `IamClient` al `listUsers` método. Puedes recuperar la lista de usuarios llamando `users` al [ListUsersResponse](#) objeto devuelto.

La lista de usuarios devuelta por `listUsers` está paginada. Puede comprobar que no haya más resultados que recuperar llamando al método `isTruncated` del objeto de respuesta. Si devuelve `true`, llame al método `marker()` del objeto de respuesta. Utilice el valor del marcador para crear un nuevo objeto de solicitud. Llame entonces al método `listUsers` de nuevo con la nueva solicitud.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Actualizar un usuario

Para actualizar un usuario, llama al `updateUser` método del `IamClient` objeto, que toma un [UpdateUserRequest](#) objeto que puedes usar para cambiar el nombre o la ruta del usuario.

Importaciones

```
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);
    }
}
```

Consulta el [ejemplo completo](#) en GitHub.

Eliminación de un usuario

Para eliminar un usuario, llama a `IamClient` la `deleteUser` solicitud con un conjunto de [UpdateUserRequest](#) objetos con el nombre de usuario que deseas eliminar.

Importaciones

```
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);
    }
}
```

Consulta el [ejemplo completo](#) en GitHub.

Más información

- [Usuarios de IAM](#) en la Guía del usuario de IAM
- [Gestión de usuarios de IAM](#) en la Guía del usuario de IAM
- [CreateUser](#) en la referencia IAM de la API
- [ListUsers](#) en la referencia IAM de la API
- [UpdateUser](#) en la referencia IAM de la API
- [DeleteUser](#) en la referencia IAM de la API

Cree políticas de IAM con AWS SDK for Java 2.x

La [API IAM Policy Builder](#) es una biblioteca que puede utilizar para crear [políticas de IAM en Java y cargarlas](#) en AWS Identity and Access Management (IAM).

En lugar de crear una política de IAM mediante el ensamblaje manual de una cadena JSON o la lectura de un archivo, la API proporciona un enfoque del cliente y orientado a los objetos para generar la cadena JSON. Al leer una política de IAM existente en formato JSON, la API la convierte en una [IamPolicy](#) instancia para su gestión.

La API Policy Builder de IAM comenzó a estar disponible con la versión 2.20.105 del SDK, así que utilice esa versión o una posterior en su archivo de compilación de Maven. El número de versión más reciente del SDK [aparece en Maven Central](#).

En el fragmento siguiente, se muestra un bloque de dependencia de ejemplo para un archivo pom.xml Maven. Esto le permite utilizar la API del creador de políticas de IAM en su proyecto.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>iam-policy-builder</artifactId>
  <version>2.20.139</version>
</dependency>
```

Crear un **IamPolicy**

En esta sección se muestran varios ejemplos de cómo crear políticas mediante la API del creador de políticas de IAM.

En cada uno de los ejemplos siguientes, comience con el [IamPolicy.Builder](#) y añada una o más instrucciones mediante el método `addStatement`. Siguiendo este patrón, el [IamStatement.Builder](#) tiene métodos para añadir el efecto, las acciones, los recursos y las condiciones a la declaración.

Ejemplo: crear una política basada en el tiempo.

El siguiente ejemplo crea una política basada en la identidad que permite la acción `GetItem` de Amazon DynamoDB entre dos momentos.

```
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());
}
```

Salida de JSON

La última instrucción del ejemplo anterior devuelve la siguiente cadena JSON.

Para obtener más información sobre este [ejemplo](#), consulte la Guía del usuario de 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"
      }
    }
  }
}
```

Ejemplo: especificar varias condiciones

En el ejemplo siguiente se muestra cómo crear una política basada en identidad que permita el acceso a los atributos específicos de DynamoDB. La política contiene dos condiciones.

```
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());
    }

```

Salida de JSON

La última instrucción del ejemplo anterior devuelve la siguiente cadena JSON.

Para obtener más información sobre este [ejemplo](#) consulte la Guía del usuario de 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"
      }
    }
  }
}

```

```
}

```

Ejemplo: especificar entidades principales

En el siguiente ejemplo, se muestra cómo crear una política basada en recursos que deniegue el acceso a un bucket a todas las entidades principales, excepto a las especificadas en la condición.

```
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());
}
```

Salida de JSON

La última instrucción del ejemplo anterior devuelve la siguiente cadena JSON.

Para obtener más información sobre este [ejemplo](#) consulte la Guía del usuario de 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"
      }
    }
  }
}
```

```

}
}

```

Ejemplo: permitir el acceso entre cuentas

En el siguiente ejemplo, se muestra cómo permitir que otra persona Cuenta de AWS cargue objetos en tu bucket y, al mismo tiempo, mantener el control total del propietario de los objetos cargados.

```

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());
}

```

Salida de JSON

La última instrucción del ejemplo anterior devuelve la siguiente cadena JSON.

Para más información sobre este [ejemplo](#), consulte la Guía del usuario de 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"
      }
    }
  }
}

```



```

    }
  }
}
}

```

Utilizar una **IamPolicy** con IAM

Tras crear una instancia de `IamPolicy`, utilice un [IamClient](#) para trabajar con el servicio de IAM.

El ejemplo siguiente crea una política que permite a una [identidad de IAM](#) escribir elementos en una tabla de DynamoDB de la cuenta que se especifica con el parámetro `accountID`. A continuación, la política se carga en IAM como una cadena 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());
}

```

El ejemplo siguiente se basa en el anterior. El código descarga la política y la utiliza como base para una nueva política copiando y modificando la declaración. A continuación, se carga la nueva política.

```

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

Los ejemplos anteriores utilizan un argumento `IamClient` que se crea tal y como se muestra en el siguiente fragmento.

```
IamClient iam = IamClient.builder().region(Region.AWS_GLOBAL).build();
```

Políticas en JSON

Los ejemplos devuelven las siguientes cadenas 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"
  }
}

```

Trabajar con políticas de IAM

Crear una política.

Para crear una nueva política, proporcione el nombre de la política y un documento de política con formato JSON en el `IamClient` método [CreatePolicyRequest](#). `createPolicy`

Importaciones

```

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 "" ;
}
```

Consulte el ejemplo [completo](#) en GitHub

Obtención de una política

Para recuperar una política existente, llame al `getPolicy` método `IamClient`'s y proporcione el ARN de la política dentro de un [GetPolicyRequest](#) objeto.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub

Asociar una política de rol

Puede adjuntar una política a un IAM [rol](#) llamando al `attachRolePolicy` método `IamClient`'s y proporcionándole el nombre del rol y el ARN de la política en un [AttachRolePolicyRequest](#)

Importaciones

```
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");
}
```

Consulte el [ejemplo completo](#) en GitHub

Mostrar las políticas de rol asociadas

Enumere las políticas adjuntas a un rol mediante una llamada al `IamClient`

`listAttachedRolePolicies` método. Se necesita un [ListAttachedRolePoliciesRequest](#) objeto que contiene el nombre del rol para enumerar las políticas.

Llame `getAttachedPolicies` al [ListAttachedRolePoliciesResponse](#) objeto devuelto para obtener la lista de políticas adjuntas. Los resultados pueden aparecer truncados; si el método `isTruncated` del objeto `ListAttachedRolePoliciesResponse` devuelve `true`, llame al método `marker` del objeto `ListAttachedRolePoliciesResponse`. Utilice el marcador devuelto para crear una nueva solicitud y use esta solicitud para llamar de nuevo a `listAttachedRolePolicies` para obtener el siguiente lote de resultados.

Importaciones

```
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");
}
```

Consulte el [ejemplo completo](#) en GitHub.

Desasociar una política de rol

Para separar una política de un rol, llame al `detachRolePolicy` método `IamClient`'s y suministre el nombre del rol y el ARN de la política en un [DetachRolePolicyRequest](#)

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub

Más información

- [Información general sobre políticas de IAM](#) en la Guía del usuario de IAM.
- [Referencia de políticas de AWS IAM](#) en la Guía del usuario de IAM.
- [CreatePolicy](#) en la referencia IAM de la API
- [GetPolicy](#) en la referencia IAM de la API
- [AttachRolePolicy](#) en la referencia IAM de la API
- [ListAttachedRolePolicies](#) en la referencia IAM de la API
- [DetachRolePolicy](#) en la referencia IAM de la API

Trabajar con certificados IAM de servidor

Para habilitar las conexiones HTTPS a su sitio web o aplicación AWS, necesita un certificado de servidor SSL/TLS. Puede utilizar un certificado de servidor proporcionado AWS Certificate Manager o obtenido de un proveedor externo.

Le recomendamos que lo utilice ACM para aprovisionar, administrar e implementar sus certificados de servidor. Con él ACM puede solicitar un certificado, implementarlo en sus AWS recursos y ACM encargarse de las renovaciones de los certificados por usted. Los certificados proporcionados por ACM son gratuitos. Para obtener más información al respecto ACM, consulte la [Guía AWS Certificate Manager del usuario](#).

Obtener un certificado de servidor

Para recuperar un certificado de servidor, llame al `getServerCertificate` método `IamClient` es y páselo [GetServerCertificateRequest](#) con el nombre del certificado.

Importaciones

```
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().getCertificateBody());

    } catch (IamException e) {
        System.err.println(e.getAwsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Consulta el [ejemplo completo](#) en GitHub.

Elaborar listas de certificados de servidor

Para ver una lista de los certificados de su servidor, llame al `listServerCertificates` método `IamClient`'s con un [ListServerCertificatesRequest](#). Este método devuelve un objeto [ListServerCertificatesResponse](#).

Llama al `serverCertificateMetadataList` método del `ListServerCertificateResponse` objeto devuelto para obtener una lista de [ServerCertificateMetadata](#) objetos que puedes usar para obtener información sobre cada certificado.

Los resultados pueden aparecer truncados; si el método `ListServerCertificateResponse` del objeto `isTruncated` devuelve `true`, llame al método `ListServerCertificatesResponse` del objeto `marker` y use el marcador para crear una nueva solicitud. Utilice la nueva solicitud para llamar de nuevo a `listServerCertificates` para obtener el siguiente lote de resultados.

Importaciones

```
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);
}
}
```

Consulte el [ejemplo completo](#) en GitHub.

Actualizar un certificado de servidor

Puede actualizar el nombre o la ruta de un certificado IAM de servidor

llamando al `updateServerCertificate` método. Se necesita un conjunto de

[UpdateServerCertificateRequest](#) objetos con el nombre actual del certificado de servidor y un nombre nuevo o una nueva ruta para su uso.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Eliminar un certificado de servidor

Para eliminar un certificado de servidor, llame al `deleteServerCertificate` método `IamClient`'s [DeleteServerCertificateRequest](#) que contenga el nombre del certificado.

Importaciones

```
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);
}
}
```

Consulte el [ejemplo completo](#) en GitHub.

Más información

- [Cómo trabajar con certificados de servidor](#) en la Guía IAM del usuario
- [GetServerCertificate](#) en la referencia IAM de la API
- [ListServerCertificates](#) en la referencia IAM de la API
- [UpdateServerCertificate](#) en la referencia IAM de la API
- [DeleteServerCertificate](#) en la referencia IAM de la API
- [AWS Certificate Manager Guía del usuario](#)

Trabaja con Kinesis

En esta sección se proporcionan ejemplos de programación [Amazon Kinesis](#) con la versión AWS SDK for Java 2.x.

Para obtener más información al respecto Kinesis, consulte la [Guía para Amazon Kinesis desarrolladores](#).

Los siguientes ejemplos incluyen únicamente el código necesario para demostrar cada técnica. El [código de ejemplo completo está disponible en GitHub](#). Desde allí, puede descargar un único archivo de código fuente o clonar el repositorio localmente para obtener todos los ejemplos para compilarlos y ejecutarlos.

Temas

- [Suscribirse a Amazon Kinesis Data Streams](#)

Suscribirse a Amazon Kinesis Data Streams

Los siguientes ejemplos muestran cómo recuperar y procesar datos de flujos de datos de Amazon Kinesis con el método `subscribeToShard`. Kinesis Data Streams ahora emplea la característica de distribución ramificada mejorada y una API de recuperación de datos de HTTP/2 de baja latencia, que permite a los desarrolladores ejecutar fácilmente varias aplicaciones de baja latencia y de alto desempeño en la misma secuencia de datos de Kinesis.

Configuración

En primer lugar, cree un cliente de Kinesis asíncrono y un objeto [SubscribeToShardRequest](#). Estos objetos se utilizan en cada uno de los siguientes ejemplos para suscribirse a eventos de Kinesis.

Importaciones

```
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 la interfaz del creador

Puede utilizar el método `builder` para simplificar la creación de [SubscribeToShardResponseHandler](#).

Con el creador, puede configurar cada devolución de llamada de ciclo de vida con una llamada de método en lugar de implementar la interfaz 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 tener más control sobre publicador, puede usar el método `publisherTransformer` para personalizar el 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);
}

```

Consulte el [ejemplo completo](#) en GitHub.

Usa un controlador de respuesta personalizado

Para tener un control completo del suscriptor y el publicador, implemente la interfaz de `SubscribeToShardResponseHandler`.

En este ejemplo, se implementa el método `onEventStream`, lo que le permite el acceso completo al publicador. Esto demuestra cómo transformar el publicador en registros de eventos para su impresión por parte del suscriptor.

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);
}

```

Consulte el [ejemplo completo](#) en GitHub.

Usa la interfaz de visitante

Puede utilizar un objeto [Visitor](#) para suscribirse a eventos específicos que le interese ver.

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);
}
```

Consulte el [ejemplo completo](#) en GitHub.

Usa un suscriptor personalizado

También puede implementar su propio suscriptor personalizado para suscribirse a la secuencia.

Este fragmento de código muestra un ejemplo de suscriptor.

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");
}
}
```

Puede pasar el suscriptor personalizado al método `subscribe`, como se muestra en el siguiente fragmento de código.

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);
}
```

Consulte el [ejemplo completo](#) en GitHub.

Escribir registros de datos en una secuencia de datos de Kinesis

Puede utilizar el objeto [KinesisClient](#) para escribir registros de datos en una secuencia de datos de Kinesis mediante el método `putRecords`. Para invocar correctamente este método, cree un objeto [PutRecordsRequest](#). Debe pasar el nombre de la secuencia de datos al método `streamName`. También debe pasar los datos mediante el método `putRecords` (como se muestra en el siguiente ejemplo de código).

Importaciones

```
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;
```

En el siguiente ejemplo de código Java, observe que el objeto `StockTrade` se utiliza como los datos para escribir en la secuencia de datos de Kinesis. Antes de ejecutar este ejemplo, asegúrese de haber creado la secuencia de datos.

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);
    }
}
}
```

Consulte el [ejemplo completo](#) en GitHub.

Usar una biblioteca de terceros

Puede utilizar otras bibliotecas de terceros en lugar de implementar un suscriptor personalizado. Este ejemplo muestra el uso de la implementación RxJava, pero puede utilizar cualquier biblioteca que implemente las interfaces de secuencias reactivas. Consulte la [página wiki de RxJava en Github](#) para obtener más información acerca de dicha biblioteca.

Para utilizar la biblioteca, añádala como una dependencia. Si está utilizando Maven, el ejemplo muestra el fragmento de POM que se debe utilizar.

Entrada de POM

```
<dependency>
  <groupId>io.reactivex.rxjava2</groupId>
  <artifactId>rxjava</artifactId>
  <version>2.1.14</version>
</dependency>
```

Importaciones

```
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;
```

En este ejemplo se utiliza RxJava en el método de ciclo de vida `onEventStream`. Esto le ofrece acceso completo al publicador, que se puede utilizar para crear un 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();
```

También puede usar el método `publisherTransformer` con el publicador `Flowable`. Debe adaptar el publicador `Flowable` a un `SdkPublisher`, tal y como se muestra en el siguiente ejemplo.

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();
```

Consulte el [ejemplo completo](#) en GitHub.

Más información

- [SubscribeToShardEvent](#) en la referencia de la API de Amazon Kinesis
- [SubscribeToShard](#) en la referencia de la API de Amazon Kinesis

Invocar, enumerar y eliminar funciones AWS Lambda

En esta sección se proporcionan ejemplos de programación con el cliente Lambda de servicio mediante la versión AWS SDK for Java 2.x.

Temas

- [Invocar una función Lambda](#)
- [Enumerar funciones de Lambda](#)
- [Eliminación de una función de Lambda](#)

Invocar una función Lambda

Para invocar una Lambda función, cree un [LambdaClient](#) objeto e invoque su método. `invoke` Crea un [InvokeRequest](#) objeto para especificar información adicional, como el nombre de la función y la carga útil que se transferirá a la función. Lambda Los nombres de las funciones aparecen como `arn:aws:lambda:us-east-1:123456789012:function:. HelloFunction` Puede recuperar el valor viendo la función en la AWS Management Console.

Para pasar los datos de la carga útil a una función, cree un objeto que contenga información. [SdkBytes](#) Por ejemplo, en el siguiente ejemplo de código, observe los datos JSON pasados a la función Lambda .

Importaciones

```
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

El siguiente ejemplo de código muestra cómo invocar una Lambda función.

```
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);
}
}
```

Consulte el [ejemplo completo](#) en. GitHub

Enumerar funciones de Lambda

Construye un [Lambda Client](#) objeto e invoca su `listFunctions` método. Este método devuelve un [ListFunctionsResponse](#) objeto. Puede invocar el `functions` método de este objeto para devolver una lista de [FunctionConfiguration](#) objetos. Puede recorrer la lista en iteración para recuperar información sobre las funciones. Por ejemplo, el siguiente ejemplo de código Java muestra cómo obtener el nombre de cada función.

Importaciones

```
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

El siguiente ejemplo de código Java muestra cómo recuperar una lista de nombres de funciones de .

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub

Eliminación de una función de Lambda

Construye un [LambdaClient](#) objeto e invoca su `deleteFunction` método. Crea un [DeleteFunctionRequest](#) objeto y pásalo al `deleteFunction` método. Este objeto contiene información como el nombre de la función que se va a eliminar. Los nombres de las funciones aparecen como `arn:aws:lambda:us-east-1:123456789012:function::HelloFunction`. Puede recuperar el valor viendo la función en la AWS Management Console.

Importaciones

```
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

El siguiente código de Java muestra cómo eliminar una función. Lambda

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Trabajo con Amazon S3

En esta sección se proporcionan ejemplos de programación con [Amazon Simple Storage Service \(S3\)](#) mediante AWS SDK for Java 2.x.

Los siguientes ejemplos incluyen únicamente el código necesario para demostrar cada técnica. El [código de ejemplo completo está disponible en GitHub](#). Desde allí, puede descargar un único archivo de código fuente o clonar el repositorio localmente para obtener todos los ejemplos para compilarlos y ejecutarlos.

Note

A partir de la versión 2.18.x, AWS SDK for Java 2.x utiliza un direccionamiento similar al de un host virtual al incluir una anulación de punto final. Esto se aplica siempre que el nombre del bucket sea una etiqueta DNS válida.

Llame al método [forcePathStyle](#) con `true` en su constructor de clientes para forzar al cliente a utilizar el direccionamiento estilo ruta para los buckets.

En el siguiente ejemplo se muestra un cliente de servicio configurado con una anulación de punto de conexión y utilizando direccionamiento estilo ruta.

```
S3Client client = S3Client.builder()
    .region(Region.US_WEST_2)
    .endpointOverride(URI.create("https://s3.us-
west-2.amazonaws.com"))
    .forcePathStyle(true)
    .build();
```

Uso de puntos de acceso o puntos de acceso de varias regiones

Una vez configurados los [puntos de acceso de Amazon S3](#) o los [puntos de acceso de varias regiones](#), puede llamar a métodos de objetos, como `putObject` y `getObject`, y proporcionar el identificador del punto de acceso en lugar de un nombre de bucket.

Por ejemplo, si el identificador ARN de un punto de acceso es `arn:aws:s3:us-west-2:123456789012:accesspoint/test`, puede usar el siguiente fragmento para llamar al 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);
```

En lugar de la cadena ARN, también puede utilizar el [alias tipo bucket](#) del punto de acceso para el parámetro `bucket`.

Para usar el punto de acceso de varias regiones, sustituya el parámetro `bucket` por el ARN del punto de acceso de varias regiones que tenga el siguiente formato.

```
arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias
```

Agregue la siguiente dependencia de Maven para trabajar con puntos de acceso de varias regiones mediante el SDK para Java. Busque en maven central la [última versión](#).

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>auth-crt</artifactId>
  <version>VERSION</version>
</dependency>
```

Temas

- [Crear, enumerar y eliminar buckets de Amazon S3](#)
- [Uso de objetos de Amazon S3](#)
- [Trabajo con URL prefiradas de Amazon S3](#)

- [Acceso entre regiones de Amazon S3](#)
- [Sumas de comprobación de Amazon S3 con AWS SDK for Java](#)
- [Utilizar un cliente S3 de alto rendimiento: cliente S3 basado en CRT AWS](#)
- [Transferir archivos y directorios con Amazon S3 Transfer Manager](#)

Crear, enumerar y eliminar buckets de Amazon S3

Cada objeto (archivo) de Amazon S3 debe residir en un bucket. Un bucket representa una colección (contenedor) de objetos. Cada bucket debe tener una clave (nombre) única. Para obtener información detallada acerca de los buckets y su configuración, consulte [Uso de buckets de Amazon S3](#) en la Guía del usuario de Amazon Simple Storage Service.

Note

Práctica recomendada

Le recomendamos que habilite la regla del ciclo de vida [AbortIncompleteMultipartUpload](#) en los buckets de Amazon S3.

Esta regla le indica a Amazon S3 que anule las cargas multiparte que no se completen en un número especificado de días después de iniciarse. Cuando se supera el plazo establecido, Amazon S3 anula la carga y, a continuación, elimina la carga de datos incompleta.

Para obtener más información, consulte [Configuración de ciclo de vida para un bucket con control de versiones](#) en la Guía del usuario de Amazon Simple Storage Service.

Note

Estos fragmentos de código presuponen que comprende los conceptos básicos y que ha configurado las credenciales de AWS predeterminadas utilizando la información de [the section called “Configuración para el acceso mediante inicio de sesión único para el SDK”](#).

Crear un bucket

Cree un objeto [CreateBucketRequest](#) y proporcione un nombre de bucket. Páselo al método `createBucket` de `S3Client`. Utilice `S3Client` para realizar operaciones adicionales como mostrar o eliminar buckets, tal y como se muestra en posteriores ejemplos.

Importaciones

```
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

Primero, cree un S3Client.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Realice una solicitud de creación 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);
        }
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Obtener una lista de buckets

Cree una [ListBucketsRequest](#). Utilice el método `listBuckets` de `S3Client` para recuperar la lista de buckets. Si la solicitud se realiza correctamente, se devuelve un objeto [ListBucketsResponse](#). Utilice este objeto de respuesta para recuperar la lista de buckets.

Importaciones

```
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

Primero, cree un `S3Client`.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Realice una solicitud de lista de buckets.

```
// List buckets
ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder().build();
ListBucketsResponse listBucketsResponse = s3.listBuckets(listBucketsRequest);
listBucketsResponse.buckets().stream().forEach(x ->
System.out.println(x.name()));
```

Consulte el [ejemplo completo](#) en GitHub.

Eliminación de un bucket

Antes de eliminar un bucket de Amazon S3, debe asegurarse de que el bucket está vacío o el servicio producirá un error. Si tiene un [bucket con control de versiones](#), también debe eliminar todos los objetos con control de versiones que están en el bucket.

Temas

- [Eliminación de objetos de un bucket](#)
- [Eliminar un bucket vacío](#)

Eliminación de objetos de un bucket

Cree un objeto [ListObjectsV2Request](#) y utilice el método `listObjects` de `S3Client` para recuperar la lista de objetos del bucket. A continuación, utilice el método `deleteObject` en cada objeto para eliminarlo.

Importaciones

```
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

Primero, cree un `S3Client`.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Elimine todos los objetos del 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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Eliminar un bucket vacío

Cree un objeto [DeleteBucketRequest](#) con un nombre de bucket y páselo al método `deleteBucket` de `S3Client`.

Importaciones

```
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

Primero, cree un S3Client.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Elimine el bucket.

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

Consulte el [ejemplo completo](#) en GitHub.

Uso de objetos de Amazon S3

Un objeto de Amazon S3 representa un archivo o conjunto de datos. Cada objeto debe estar incluido en un [bucket](#).

Note

Práctica recomendada

Le recomendamos que habilite la regla del ciclo de vida [AbortIncompleteMultipartUpload](#) en los buckets de Amazon S3.

Esta regla le indica a Amazon S3 que anule las cargas multiparte que no se completen en un número especificado de días después de iniciarse. Cuando se supera el plazo establecido, Amazon S3 anula la carga y, a continuación, elimina la carga de datos incompleta.

Para obtener más información, consulte [Configuración de ciclo de vida para un bucket con control de versiones](#) en la Guía del usuario de Amazon Simple Storage Service.

Note

Estos fragmentos de código presuponen que comprende los conceptos básicos y que ha configurado las credenciales de AWS predeterminadas utilizando la información de [the section called “Configuración para el acceso mediante inicio de sesión único para el SDK”](#).

Temas

- [Cargar un objeto](#)
- [Cargar objetos en varias partes](#)
- [Elimine un objeto](#)
- [List objects](#)
- [Más ejemplos](#)

Cargar un objeto

Cree un objeto [PutObjectRequest](#) y proporcione un nombre de bucket y un nombre de clave. A continuación, utilice el método `putObject` de `S3Client` con un [RequestBody](#) que incluya el contenido del objeto y el objeto `PutObjectRequest`. El bucket debe existir o el servicio producirá un error.

Importaciones

```
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)));
```

Consulte el [ejemplo completo](#) en GitHub.

Cargar objetos en varias partes

Utilice el método `createMultipartUpload` de `S3Client` para obtener un ID de carga. A continuación, use el método `uploadPart` para cargar cada parte. Por último, utilice el método `completeMultipartUpload` de `S3Client` para indicar a Amazon S3 que combine todas las partes cargadas y finalice la operación de carga.

Importaciones

```
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);
```

Consulte el [ejemplo completo](#) en GitHub.

Elimine un objeto

Cree un objeto [DeleteObjectRequest](#) y proporcione un nombre de bucket y un nombre de clave. Utilice el método `deleteObject` de `S3Client` y pase el nombre del bucket y el objeto que se van a eliminar. El bucket y la clave de objeto especificados deben existir o el servicio producirá un error.

Importaciones

```
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);
```

Consulte el [ejemplo completo](#) en GitHub.

Copiar un objeto

Cree un [CopyObjectRequest](#) y proporcione un nombre de bucket en el que se aplica el objeto, un valor de cadena codificado en URL (consulte el método `URLEncoder.encode`) y el nombre de clave del objeto. Utilice el método `copyObject` de `S3Client` y pase el objeto [CopyObjectRequest](#). El bucket y la clave de objeto especificados deben existir o el servicio producirá un error.

Importaciones

```
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 "";
}
}
```

Consulte el [ejemplo completo](#) en GitHub.

List objects

Cree una [ListObjectsRequest](#) y proporcione el nombre del bucket. A continuación, invoque el método `listObjects` de `S3Client` y pase el objeto `ListObjectsRequest`. Este método devuelve un [ListObjectsResponse](#) que contiene todos los objetos del bucket. Puede invocar el método de contenido de este objeto para obtener una lista de objetos. Puede recorrer esta lista para mostrar los objetos, como se muestra en el ejemplo de código siguiente.

Importaciones

```
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.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 calKb(Long val) {
    return val / 1024;
}
}
```

Consulte el [ejemplo completo](#) en GitHub.

Más ejemplos

La sección [Ejemplos de código](#) de esta guía contiene más ejemplos de cómo trabajar con objetos de Amazon S3, incluida la forma de [descargar un objeto](#).

Trabajo con URL prefiradas de Amazon S3

Las URL prefiradas proporcionan acceso temporal a objetos privados de S3 sin necesidad de que los usuarios tengan credenciales o permisos de AWS.

Por ejemplo, supongamos que Alice tiene acceso a un objeto S3 y desea compartir temporalmente el acceso a ese objeto con Bob. Alice puede generar una solicitud GET prefirada para compartir con Bob de forma que pueda descargar el objeto sin requerir acceso a las credenciales de Alice. Puede generar direcciones URL prefiradas para solicitudes HTTP GET y HTTP PUT.

Genere una URL prefirada para un objeto y, a continuación, descárguela (solicitud GET)

El siguiente ejemplo consta de dos partes.

- Parte 1: Alice genera la URL prefirada de un objeto.
- Parte 2: Bob descarga el objeto mediante la URL prefirada.

Parte 1: Generar la URL

Alice ya tiene un objeto en un bucket S3. Usa el siguiente código para generar una cadena URL que Bob puede usar en una solicitud GET posterior.

Importaciones

```
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: Descargar el objeto

Bob usa una de las siguientes tres opciones de código para descargar el objeto. Como alternativa, podría usar un navegador para realizar la solicitud GET.

Usar JDK **HttpURLConnection** (desde la versión 1.1)

```

/* 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 JDK `HttpClient` (desde la 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` desde el 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();
    }
}
```

Consulte el [ejemplo completo](#) y la [prueba](#) en GitHub.

Genere una URL prefirmada para una carga y, a continuación, cargue un archivo (solicitud PUT)

El siguiente ejemplo consta de dos partes.

- Parte 1: Alice genera la URL prefirmada para cargar un objeto.
- Parte 2: Bob carga un archivo mediante la URL prefirmada.

Parte 1: Generar la URL

Alice ya tiene un bucket de S3. Usa el siguiente código para generar una cadena URL que Bob puede usar en una solicitud PUT posterior.

Importaciones

```
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: Cargar un objeto de archivo

Bob usa una de las siguientes tres opciones de código para cargar un archivo.

Usar JDK **URLConnection** (desde la versión 1.1)

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useURLConnectionToPut(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 JDK `HttpClient` (desde la 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` desde el 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);
    }
}
}

```


Consulte el [ejemplo completo](#) y la [prueba](#) en GitHub.

Acceso entre regiones de Amazon S3

Cuando trabaja con buckets de Amazon Simple Storage Service (Amazon S3), normalmente sabe cuál es la Región de AWS del bucket. La región con la que trabaja se determina al crear el cliente S3.

Sin embargo, a veces necesitará trabajar con un bucket específico, pero no sabe si se encuentra en la misma región que está configurada para el cliente S3.

En lugar de hacer más llamadas para determinar la región del bucket, puede usar el SDK para habilitar el acceso a los buckets de S3 en distintas regiones.

Configuración

El acceso entre regiones comenzó a estar disponible con la versión 2.20.111 del SDK. Use esta versión o una posterior en su archivo de compilación de Maven para la dependencia `s3`, como se muestra en el siguiente fragmento.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.111</version>
</dependency>
```

Después, cuando cree su cliente S3, habilite el acceso entre regiones, como se muestra en el fragmento. El acceso no está habilitado de forma predeterminada.

```
S3AsyncClient client = S3AsyncClient.builder()
    .crossRegionAccessEnabled(true)
    .build();
```

Cómo proporciona el SDK acceso entre regiones

Al hacer referencia a un bucket existente en una solicitud, como cuando utiliza el método `putObject`, el SDK inicia una solicitud a la Región configurada para el cliente.

Si el bucket no existe en esa región específica, la respuesta al error incluye la región real en la que reside el bucket. A continuación, el SDK utiliza la región correcta en una segunda solicitud.

Para optimizar las futuras solicitudes al mismo bucket, el SDK almacena en caché esta asignación de regiones en el cliente.

Consideraciones

Cuando active el acceso a los buckets entre regiones, tenga en cuenta que la primera llamada a la API puede provocar un aumento de la latencia si el bucket no se encuentra en la región configurada del cliente. Sin embargo, las llamadas posteriores se benefician de la información regional almacenada en caché, lo que mejora el rendimiento.

Al habilitar el acceso entre regiones, el acceso al bucket no se ve afectado. El usuario debe estar autorizado a acceder al bucket en cualquier región en la que resida.

Sumas de comprobación de Amazon S3 con AWS SDK for Java

Amazon Simple Storage Service (Amazon S3) permite especificar una suma de comprobación al cargar un objeto. Cuando se especifica una suma de comprobación, esta se almacena con el objeto y se puede validar cuando se descarga el objeto.

Las sumas de comprobación proporcionan un nivel adicional de integridad de los datos al transferir archivos. Con las sumas de comprobación, puede comprobar la coherencia de datos verificando que el archivo recibido coincide con el archivo original. Para obtener más información acerca de las sumas de comprobación con Amazon S3, consulte la [guía del usuario de Amazon Simple Storage Service](#).

Amazon S3 admite actualmente cuatro algoritmos de suma de comprobación: SHA-1, SHA-256, CRC-32 y CRC-32C. Puede elegir el algoritmo que mejor se adapte a sus necesidades y dejar que el SDK calcule la suma de comprobación. También puede especificar un valor de suma de comprobación calculado previamente mediante uno de los cuatro algoritmos compatibles.

Analizaremos las sumas de comprobación en dos fases de solicitud: carga del objeto y descarga del objeto.

Cargar un objeto

Los valores válidos del algoritmo son CRC32, CRC32C, SHA1 y SHA256.

El siguiente fragmento de código muestra una solicitud para cargar un objeto con una suma de comprobación de CRC-32. Cuando el SDK envía la solicitud, calcula la suma de comprobación de CRC-32 y carga el objeto. Amazon S3 almacena la suma de comprobación en el objeto.

Si la suma de comprobación que calcula el SDK no coincide con la suma de comprobación que calcula Amazon S3 al recibir la solicitud, se devuelve un error.

Utilizar un valor de suma de comprobación calculado previamente

Un valor de suma de comprobación precalculado proporcionado con la solicitud desactiva el cálculo automático por parte del SDK y utiliza el valor proporcionado en su lugar.

En el siguiente ejemplo se muestra una solicitud con una suma de comprobación SHA-256 precalculada.

Si Amazon S3 determina que el valor de la suma de comprobación es incorrecto para el algoritmo especificado, el servicio devuelve una respuesta de error.

Cargas multiparte

También puede utilizar sumas de comprobación en las cargas multiparte.

Descargar un objeto

Cuando se utiliza el método [getObject](#) para descargar un objeto, el SDK valida automáticamente la suma de comprobación

La solicitud del siguiente fragmento indica al SDK que valide la suma de comprobación de la respuesta calculándola y comparando los valores.

Si el objeto no se cargó con una suma de comprobación, no se realizará ninguna validación.

Un objeto de Amazon S3 puede tener varias sumas de comprobación, pero solo se valida una de ellas al descargarse. La siguiente prioridad, basada en la eficacia del algoritmo de suma de comprobación, determina qué suma de comprobación valida el SDK:

1. CRC-32C
2. CRC-32
3. SHA-1
4. SHA-256

Por ejemplo, si una respuesta contiene las sumas de comprobación CRC-32 y SHA-256, solo se valida la de CRC-32.

Utilizar un cliente S3 de alto rendimiento: cliente S3 basado en CRT AWS

El cliente S3 basado en AWS CRT, creado sobre el [AWSCommon Runtime \(CRT\)](#), es un cliente asíncrono S3 alternativo. Transfiere objetos hacia y desde Amazon Simple Storage Service (Amazon S3) con un rendimiento y una fiabilidad mejorados mediante el uso automático de la API de [carga multiparte de Amazon S3](#) y de las [recuperaciones por rango de bytes](#).

El cliente S3 basado en AWS CRT mejora la fiabilidad de la transferencia en caso de que se produzca un fallo en la red. La fiabilidad se mejora reintentando partes individuales fallidas de una transferencia de archivos sin reiniciar la transferencia desde el principio.

Además, el cliente S3 basado en AWS CRT ofrece una mejor agrupación de conexiones y un equilibrio de carga del Sistema de Nombres de Dominio (DNS) mejorados, lo que también mejora el rendimiento.

Puede utilizar el cliente S3 basado en AWS CRT en lugar del cliente asíncrono S3 estándar del SDK y aprovechar su rendimiento mejorado de forma inmediata.

Componentes basados en AWS CRT en el SDK

El cliente S3 basado en AWS CRT, que se describe en este tema, y el cliente HTTP basado en AWS CRT son componentes diferentes del SDK.

El cliente S3 AWS basado en CRT es una implementación de la interfaz [S3AsyncClient](#) y se utiliza para trabajar con el servicio Amazon S3. Es una alternativa a la implementación de la interfaz `S3AsyncClient` basada en Java y ofrece varias ventajas.

El [cliente HTTP basado en AWS CRT](#) es una implementación de la interfaz [SDKAsynchTTPClient](#) y se utiliza para la comunicación HTTP general. Es una alternativa a la implementación Netty de la interfaz `SdkAsynchHttpClient` basada en Java y ofrece varias ventajas.

Aunque ambos componentes utilizan bibliotecas del [AWS Common Runtime](#), el cliente S3 basado en AWS CRT utiliza la biblioteca [aws-c-s3](#) y soporta las características de la [API de carga multiparte de S3](#). Como el cliente HTTP basado en AWS CRT está diseñado para un uso general, no admite las características de la API de carga multiparte de S3.

Agregar dependencias para usar el cliente S3 basado en AWS CRT

Para usar el cliente S3 basado en AWS CRT, añada las dos dependencias siguientes al archivo de proyecto de Maven. En el ejemplo siguiente se muestran las versiones mínimas que se utilizarán. Busque en el repositorio central de Maven las versiones más recientes de los artefactos [s3](#) y [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>
```

Crear una instancia del cliente S3 basado en AWS CRT

Cree una instancia del cliente S3 basado en AWS CRT con la configuración predeterminada, como se muestra en el siguiente fragmento de código.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
```

Para configurar el cliente, utilice el creador de clientes AWS CRT. Puede cambiar del cliente asíncrono S3 estándar al cliente basado en AWS CRT cambiando el método de creación.

```
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

Es posible que algunas de las configuraciones del creador estándar no sean compatibles actualmente con el creador de clientes AWS CRT. Obtener el creador estándar llamando a `S3AsyncClient#builder()`.

Utilizar el Cliente S3 basado en AWS CRT

Utilice el cliente S3 basado en AWS CRT para llamar a las operaciones de API de Amazon S3. El ejemplo siguiente muestra las operaciones [PutObject](#) y [GetObject](#) mediante el 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();
```

Transferir archivos y directorios con Amazon S3 Transfer Manager

Amazon S3 Transfer Manager es una utilidad de transferencia de archivos de alto nivel y de código abierto para el AWS SDK for Java 2.x. Úselo para transferir archivos y directorios desde y hacia Amazon Simple Storage Service (Amazon S3).

Cuando se construye sobre [el cliente S3 basado en CRT de AWS](#) el S3 Transfer Manager puede aprovechar las mejoras de rendimiento, como la [API de carga multiparte](#) y las [recuperaciones de rango de bytes](#).

Con el S3 Transfer Manager, también puede supervisar el progreso de una transferencia en tiempo real y pausarla para su posterior ejecución.

Introducción

Añadir dependencias a su archivo de compilación

Para utilizar el S3 Transfer Manager con un rendimiento mejorado basado en el cliente S3 AWS basado en CRT, configure el archivo de compilación con las siguientes dependencias.

- Utilice la versión **2.19.1** o superior del SDK para Java 2.x.
- Agregue el artefacto `s3-transfer-manager` como dependencia.
- Agregue el artefacto `aws-crt` como una dependencia en la versión **0.20.3** o superior.

En el ejemplo de código siguiente se muestra cómo configurar las dependencias de su proyecto para Maven.

```
<project>
  <properties>
    <aws.sdk.version>2.19.1</aws.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3-transfer-manager</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk.crt</groupId>
      <artifactId>aws-crt</artifactId>
      <version>0.20.3</version>
    </dependency>
  </dependencies>
</project>
```

Busque en el repositorio central de Maven las versiones más recientes de los artefactos [s3-transfer-manager](#) y [aws-crt](#).

Crear una instancia del S3 Transfer Manager

En el siguiente fragmento se muestra cómo crear una instancia de [S3 TransferManager](#) con la configuración predeterminada.

```
S3TransferManager transferManager = S3TransferManager.create();
```

El ejemplo siguiente muestra cómo configurar un S3 Transfer Manager con ajustes personalizados. En este ejemplo, se utiliza una AsyncClient instancia [S3 AWS basada en CRT](#) como cliente subyacente del 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

Si la dependencia `aws-crt` no está incluida en el archivo de compilación, S3 Transfer Manager se crea sobre el cliente asíncrono S3 estándar que se utiliza en el SDK para Java 2.x.

Cargar un archivo en un bucket de S3

El siguiente ejemplo muestra un ejemplo de carga de archivos junto con el uso opcional de [LoggingTransferListener](#), que registra el progreso de la carga.

Para cargar un archivo en Amazon S3 mediante el S3 Transfer Manager, pase un objeto [UploadFileRequest](#) al método [uploadFile](#) del `S3TransferManager`.

El [FileUpload](#) objeto devuelto por el `uploadFile` método representa el proceso de carga. Una vez finalizada la solicitud, el [CompletedFileUpload](#) objeto contiene información sobre la carga.

```
public String uploadFile(S3TransferManager transferManager, String bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

Importaciones

```
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;
```

Descargar un archivo de un bucket de S3

El siguiente ejemplo muestra un ejemplo de descarga junto con el uso opcional de un [LoggingTransferListener](#), que registra el progreso de la descarga.

Para descargar un objeto de un bucket de S3 mediante el administrador de transferencias de S3, cree un [DownloadFileRequest](#) objeto y páselo al método [downloadFile](#).

El [FileDownload](#) objeto devuelto por el `downloadFile` método `S3TransferManager`'s representa la transferencia de archivos. Una vez completada la descarga, [CompletedFileDownload](#) contiene el acceso a la información sobre la descarga.

```
public Long downloadFile(S3TransferManager transferManager, String bucketName,
                        String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .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();
}
```

Importaciones

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

Copiar un objeto de Amazon S3 a otro bucket

En el siguiente ejemplo se muestra cómo copiar un objeto con S3 Transfer Manager.

Para empezar a copiar un objeto de un bucket de S3 a otro bucket, cree una

[CopyObjectRequest](#) instancia básica.

A continuación, incluya la básica `CopyObjectRequest` en una [CopyRequest](#) que pueda utilizar el S3 Transfer Manager.

El objeto `Copy` devuelto por el método `copy` de `S3TransferManager` representa el proceso de copia. Una vez finalizado el proceso de copia, el [CompletedCopy](#) objeto contiene detalles sobre la respuesta.

```
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 una copia entre regiones con el administrador de transferencias de S3, habilítelo `crossRegionAccessEnabled` en el generador de clientes S3 AWS basado en CRT, tal y como se muestra en el siguiente fragmento.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .crossRegionAccessEnabled(true)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

Importaciones

```
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;
```

Cargar un directorio local en un bucket de S3

En el siguiente ejemplo se muestra cómo se puede cargar un directorio local en S3.

Comience por llamar al método [UploadDirectory](#) de la instancia e introduzca un `S3TransferManager` [UploadDirectoryRequest](#)

El [DirectoryUpload](#) objeto representa el proceso de carga, que genera un [CompletedDirectoryUpload](#) cuando se completa la solicitud. El objeto `CompletedDirectoryUpload` contiene información sobre los resultados de la transferencia, incluidos los archivos que no se pudieron transferir.

```
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();
}
```

Importaciones

```
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;
```

Descargar objetos de bucket S3 en un directorio local

Puede descargar los objetos de un bucket S3 en un directorio local tal y como se muestra en el siguiente ejemplo.

Para descargar los objetos de un bucket de S3 a un directorio local, comience por llamar al método [downloadDirectory](#) del Transfer Manager e introduzca un [DownloadDirectoryRequest](#).

El [DirectoryDownload](#) objeto representa el proceso de descarga, que genera un [CompletedDirectoryDownload](#) cuando se completa la solicitud. El objeto [CompletedDirectoryDownload](#) contiene información sobre los resultados de la transferencia, incluidos los archivos que no se pudieron transferir.

```
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();  
}
```

Importaciones

```
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;
```

Vea ejemplos completos

[GitHub contiene el código completo](#) de todos los ejemplos de esta página.

Trabajar con Amazon Simple Notification Service

Con Amazon Simple Notification Service, puede enviar fácilmente mensajes de notificación push en tiempo real desde sus aplicaciones a los suscriptores a través de distintos canales de comunicación. En este tema se describe cómo ejecutar algunas de las funciones básicas de Amazon SNS.

Crear un tema

Un tema es una agrupación lógica de canales de comunicación que define a qué sistemas se envía un mensaje (por ejemplo, la difusión de un mensaje a AWS Lambda y un webhook HTTP). Se envían mensajes a Amazon SNS y, a continuación, se distribuyen a los canales definidos en el tema. Esto permite que los mensajes estén disponibles para los suscriptores.

Para crear un tema, primero cree un objeto [CreateTopicRequest](#) con el nombre del conjunto de temas utilizando el método `name()` en el creador. A continuación, envíe el objeto de solicitud a Amazon SNS mediante el método `createTopic()` de [SnsClient](#). Puede capturar el resultado de esta solicitud como un objeto [CreateTopicResponse](#), como se muestra en el siguiente fragmento de código.

Importaciones

```
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 "";
}
```

Consulte el [ejemplo completo](#) en GitHub.

Enumerar sus temas de Amazon SNS

Para recuperar una lista de los temas de Amazon SNS existentes, cree un objeto [ListTopicsRequest](#). A continuación, envíe el objeto de solicitud a Amazon SNS mediante el método `listTopics()` de [SnsClient](#). Puede capturar el resultado de esta solicitud como un objeto [ListTopicsResponse](#).

El siguiente fragmento de código imprime el código de estado HTTP de la solicitud y una lista de nombres de recursos de Amazon (ARN) para los temas de Amazon SNS.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Suscribir un punto de enlace a un tema

Después de crear un tema, puede configurar los canales de comunicación que serán los puntos de enlace de ese tema. Los mensajes se distribuyen a estos puntos de enlace una vez que Amazon SNS los recibe.

Para configurar un canal de comunicación como punto de enlace de un tema, suscriba ese punto de enlace al tema. Para empezar, cree un objeto [SubscribeRequest](#). Especifique el canal de comunicación (por ejemplo, lambda o email) como `protocol()`. Establezca `endpoint()` en la ubicación de salida correspondiente (por ejemplo, el ARN de una función Lambda o una dirección de correo electrónico) y, a continuación, el ARN del tema al que desea suscribirse como `topicArn()`.

Envíe el objeto de solicitud a Amazon SNS mediante el método `subscribe()` del `SnsClient`. Puede capturar el resultado de esta solicitud como un objeto [SubscribeResponse](#).

El siguiente fragmento de código muestra cómo suscribir una dirección de correo electrónico a un tema.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Publicar un mensaje en un tema

Después de haber configurado un tema y uno o varios puntos de enlace para él, puede publicar un mensaje en el tema. Para empezar, cree un objeto [PublishRequest](#). Especifique el `message()`

que desea enviar y el ARN del tema (`topicArn()`) al que desea enviarlo. A continuación, envíe el objeto de solicitud a Amazon SNS mediante el método `publish()` de `SnsClient`. Puede capturar el resultado de esta solicitud como un objeto [PublishResponse](#).

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Cancelar la suscripción de un punto de enlace de un tema

Puede quitar los canales de comunicación configurados como puntos de enlace de un tema. Una vez hecho esto, el tema sigue existiendo y distribuye mensajes a cualquier otro punto de enlace configurado para ese tema.

Para quitar un canal de comunicación como punto de enlace de un tema, cancele la suscripción de dicho punto de enlace del tema. Para empezar, cree un objeto [UnsubscribeRequest](#) y establezca el

ARN del tema del que desea cancelar la suscripción como `subscriptionArn()`. A continuación, envíe el objeto de solicitud a SNS mediante el método `unsubscribe()` de `SnsClient`. Puede capturar el resultado de esta solicitud como un objeto [UnsubscribeResponse](#).

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Eliminación de un tema

Para eliminar un tema de Amazon SNS, primero cree un objeto [DeleteTopicRequest](#) con el ARN del conjunto de temas como el método `topicArn()` en el creador. A continuación, envíe el objeto de solicitud a Amazon SNS mediante el método `deleteTopic()` de `SnsClient`. Puede capturar el resultado de esta solicitud como un objeto [DeleteTopicResponse](#), como se muestra en el siguiente fragmento de código.

Importaciones

```
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);
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Para obtener más información, consulte [Amazon Simple Notification Service Developer Guide](#).

Trabaja con Amazon Simple Queue Service

En esta sección se proporcionan ejemplos de programación [Amazon Simple Queue Service](#) con la versión AWS SDK for Java 2.x.

Los siguientes ejemplos incluyen únicamente el código necesario para demostrar cada técnica. El [código de ejemplo completo está disponible en GitHub](#). Desde allí, puede descargar un único archivo de código fuente o clonar el repositorio localmente para obtener todos los ejemplos para compilarlos y ejecutarlos.

Temas

- [Uso de colas de mensajes de Amazon Simple Queue Service](#)
- [Enviar, recibir y eliminar mensajes de Amazon Simple Queue Service](#)

Uso de colas de mensajes de Amazon Simple Queue Service

Una cola de mensajes es el contenedor lógico utilizado para enviar mensajes de forma fiable en Amazon Simple Queue Service. Existen dos tipos de colas: estándar y primero en entrar, primero en salir (FIFO). Para obtener más información sobre las colas y las diferencias entre estos tipos, consulte la [Guía para desarrolladores de Amazon Simple Queue Service](#).

En este tema se describe cómo crear, mostrar, eliminar y obtener la dirección URL de un cola de Amazon Simple Queue Service mediante AWS SDK for Java.

La variable `sqsClient` que se utiliza en los ejemplos siguientes se puede crear a partir del siguiente fragmento.

```
SqsClient sqsClient = SqsClient.create();
```

Al crear una `SqsClient` mediante el `create()` método estático, el SDK configura la región mediante la cadena de [proveedores de regiones predeterminada y las credenciales mediante la cadena](#) de [proveedores de credenciales predeterminada](#).

Creación de una cola

Usa el `SqsClient`'s `createQueue` método y proporciona un [CreateQueueRequest](#) objeto que describa los parámetros de la cola, tal y como se muestra en el siguiente fragmento de código.

Importaciones

```
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);
```

Consulte el ejemplo [completo](#) en GitHub

Mostrar colas

Para ver las Amazon Simple Queue Service colas de tu cuenta, llama al `SqsClient`'s `listQueues` método con un [ListQueuesRequest](#) objeto.

Si utilizas la forma del [listQueues](#) método que no incluye parámetros, el servicio devuelve todas las colas (hasta un máximo de 1000 colas).

Puede proporcionar un prefijo de nombre de cola al [ListQueuesRequest](#) objeto para limitar los resultados a las colas que coincidan con ese prefijo, como se muestra en el código siguiente.

Importaciones

```
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);
}
```

[Consulte el ejemplo completo en GitHub](#)

Obtener la URL de una cola

El código siguiente muestra cómo obtener la URL de una cola llamando al `SqsClient`'s `getQueueUrl` método con un [GetQueueUrlRequest](#) objeto.

Importaciones

```
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;
```

Consulta el [ejemplo completo](#) en GitHub

Eliminar una cola

Proporcione la [URL](#) de la cola al [DeleteQueueRequest](#) objeto. A continuación, llama al `SqsClient`'s `deleteQueue` método para eliminar una cola, tal y como se muestra en el código siguiente.

Importaciones

```
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);  
}  
}
```

Consulta el [ejemplo completo](#) en GitHub

Más información

- [CreateQueue](#) en la referencia Amazon Simple Queue Service de la API
- [GetQueueUrl](#) en la referencia Amazon Simple Queue Service de la API
- [ListQueues](#) en la referencia Amazon Simple Queue Service de la API
- [DeleteQueue](#) en la referencia Amazon Simple Queue Service de la API

Enviar, recibir y eliminar mensajes de Amazon Simple Queue Service

Un mensaje es un segmento de datos que los componentes distribuidos pueden enviar y recibir. Los mensajes se envían siempre a través de una [cola de SQS](#).

La variable `sqsClient` que se utiliza en los ejemplos siguientes se puede crear a partir del siguiente fragmento.

```
SqsClient sqsClient = SqsClient.create();
```


Al crear una `SqsClient` mediante el `create()` método estático, el SDK configura la región mediante la cadena de [proveedores de regiones predeterminada y las credenciales mediante la cadena](#) de [proveedores de credenciales predeterminada](#).

Enviar un mensaje

Agrega un solo mensaje a una Amazon Simple Queue Service cola llamando al método `SqsClient client.sendMessage`. Proporcione un [SendMessageRequest](#) objeto que contenga la [URL](#) de la cola, el cuerpo del mensaje y el valor de retraso opcional (en segundos).

Importaciones

```
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 varios mensajes en una solicitud

Envíe varios mensajes en una sola solicitud mediante el método `SqsClient sendMessageBatch`. Este método toma una [SendMessageBatchRequest](#) que contiene la URL de la cola y una lista de los mensajes que se van a enviar. (Cada mensaje es un [SendMessageBatchRequestEntry](#).) También puede retrasar el envío de un mensaje específico estableciendo un valor de retraso en el mensaje.

Importaciones

```
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);
```

Consulte el [ejemplo completo](#) en GitHub.

Recuperar mensajes

Recupere todos los mensajes que se encuentran actualmente en la cola llamando al método `SqsClient receiveMessage`. Este método toma una [ReceiveMessageRequest](#) que contiene la URL de la cola. También puede especificar el número máximo de mensajes que se devuelven. Los mensajes se devuelven como una lista de objetos [Message](#).

Importaciones

```
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;
```

Consulte el [ejemplo completo](#) en GitHub

Eliminar un mensaje después de su recepción

Tras recibir un mensaje y procesar su contenido, elimínelo de la cola enviando el identificador de recepción del mensaje y la URL de la cola al SqsClient 's [deleteMessage](#) método.

Importaciones

```
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);
        }
    }
```

Consulta el [ejemplo completo](#) en GitHub

Más información

- [Cómo funcionan las colas de Amazon Simple Queue Service](#) en la Guía para desarrolladores de Amazon Simple Queue Service
- [SendMessage](#) en la referencia Amazon Simple Queue Service de la API
- [SendMessageBatch](#) en la referencia Amazon Simple Queue Service de la API
- [ReceiveMessage](#) en la referencia Amazon Simple Queue Service de la API

- [DeleteMessage](#) en la referencia Amazon Simple Queue Service de la API

Trabajar con Amazon Transcribe

En el siguiente ejemplo se muestra cómo funciona el streaming bidireccional con Amazon Transcribe. El streaming bidireccional implica que hay una secuencia de datos que se envía al servicio y que se recibe en tiempo real. El ejemplo utiliza transcripción de streaming de Amazon Transcribe para enviar una secuencia de audio y recibir una secuencia de texto transcrito en tiempo real.

Consulte [Transcripción de streaming](#) en la Guía para desarrollador de Amazon Transcribe para obtener más información acerca de esta característica.

Consulte [Primeros pasos](#) en la Guía para desarrolladores de Amazon Transcribe para empezar a usar Amazon Transcribe.

Configurar el micrófono

Este código utiliza el paquete `javax.sound.sampled` para transmitir audio desde un 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;
    }
}
```

Consulte el [ejemplo completo](#) en GitHub.

Crear un editor

Este código implementa un editor que publica datos de audio desde la transmisión de audio de 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();
                    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();
}
}
```

Consulte el [ejemplo completo](#) en GitHub.

Cree el cliente e inicie la secuencia

En el método principal, cree un objeto de solicitud, inicie la secuencia de entrada de audio y cree instancias del editor con la entrada de audio.

Asimismo, debe crear un [StartStreamTranscriptionResponseHandler](#) para especificar cómo gestionar la respuesta desde Amazon Transcribe.

A continuación, utilice el método `startStreamTranscription` de `TranscribeStreamingAsyncClient` para iniciar el streaming bidireccional.

Importaciones

```
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);  
    }  
}
```

Consulte el [ejemplo completo](#) en GitHub.

Más información

- [Cómo funciona](#) en la Guía para desarrolladores de Amazon Transcribe.
- [Introducción al audio en streaming](#) en la Guía para desarrolladores de Amazon Transcribe.

Ejemplos de código de AWS SDK para Java 2.x

En los ejemplos de código de este tema se muestra cómo utilizar el AWS SDK for Java 2.x with AWS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Los ejemplos entre servicios son aplicaciones de muestra que funcionan en varios Servicios de AWS.

Ejemplos

- [Acciones y escenarios con SDK para Java 2.x](#)
- [Ejemplos de servicios cruzados con SDK para Java 2.x](#)

Acciones y escenarios con SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Java 2.x with Servicios de AWS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Servicios

- [Ejemplos de puerta de enlace de API con SDK para Java 2.x](#)
- [Ejemplos de Application Auto Scaling con SDK for Java 2.x](#)
- [Ejemplos de controlador de recuperación de aplicaciones con SDK para Java 2.x](#)
- [Ejemplos de Aurora usando SDK para Java 2.x](#)

- [Ejemplos de escalado automático usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Bedrock usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Bedrock Runtime usando SDK para Java 2.x](#)
- [CloudFront ejemplos de uso de SDK for Java 2.x](#)
- [CloudWatch ejemplos de uso de SDK for Java 2.x](#)
- [CloudWatch Ejemplos de eventos con SDK for Java 2.x](#)
- [CloudWatch Ejemplos de registros mediante SDK for Java 2.x](#)
- [Ejemplos de identidad de Amazon Cognito usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Cognito Identity Provider usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Comprehend usando SDK para Java 2.x](#)
- [Ejemplos de DynamoDB usando SDK para Java 2.x](#)
- [Ejemplos de Amazon EC2 usando SDK para Java 2.x](#)
- [Ejemplos de Amazon ECR con el SDK para Java 2.x](#)
- [Ejemplos de Amazon ECS usando SDK para Java 2.x](#)
- [Ejemplos de Elastic Load Balancing: versión 2 con SDK para Java 2.x](#)
- [MediaStore ejemplos de uso de SDK for Java 2.x](#)
- [OpenSearch Ejemplos de servicios que utilizan SDK for Java 2.x](#)
- [EventBridge ejemplos de uso de SDK for Java 2.x](#)
- [Ejemplos de pronóstico usando SDK para Java 2.x](#)
- [AWS Glue ejemplos de uso de SDK for Java 2.x](#)
- [HealthImaging ejemplos de uso de SDK for Java 2.x](#)
- [Ejemplos de IAM usando SDK para Java 2.x](#)
- [AWS IoT ejemplos de uso de SDK for Java 2.x](#)
- [AWS IoT data ejemplos de uso de SDK for Java 2.x](#)
- [Ejemplos de Amazon Keyspaces usando SDK para Java 2.x](#)
- [Ejemplos de Kinesis que utilizan SDK para Java 2.x](#)
- [AWS KMS ejemplos de uso de SDK for Java 2.x](#)
- [Ejemplos de Lambda usando SDK para Java 2.x](#)
- [MediaConvert ejemplos de uso de SDK for Java 2.x](#)

- [Ejemplos de Migration Hub usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Personalize usando SDK para Java 2.x](#)
- [Ejemplos de eventos de Amazon Personalize usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Personalize Runtime usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Pinpoint usando SDK para Java 2.x](#)
- [Ejemplos de código de la API de SMS y voz de Amazon Pinpoint usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Polly usando SDK para Java 2.x](#)
- [Ejemplos de Amazon RDS usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Redshift usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Rekognition usando SDK para Java 2.x](#)
- [Ejemplos de registro de dominios de Route 53 usando SDK para Java 2.x](#)
- [Ejemplos de Amazon S3 usando SDK para Java 2.x](#)
- [Ejemplos de S3 Glacier con SDK para Java 2.x](#)
- [SageMaker ejemplos de uso de SDK for Java 2.x](#)
- [Ejemplos de Secrets Manager usando SDK para Java 2.x](#)
- [Ejemplos de Amazon SES usando SDK para Java 2.x](#)
- [Ejemplos de SES API v2 usando SDK para Java 2.x](#)
- [Ejemplos de Amazon SNS usando SDK para Java 2.x](#)
- [Ejemplos de Amazon SQS usando SDK para Java 2.x](#)
- [Ejemplos de funciones de pasos usando SDK para Java 2.x](#)
- [AWS STS ejemplos de uso de SDK for Java 2.x](#)
- [AWS Support ejemplos de uso de SDK for Java 2.x](#)
- [Ejemplos de Systems Manager usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Textract usando SDK para Java 2.x](#)
- [Ejemplos de Amazon Transcribe usando SDK para Java 2.x](#)

Ejemplos de puerta de enlace de API con SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante API Gateway.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

CreateDeployment

En el siguiente ejemplo de código, se muestra cómo usar CreateDeployment.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

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

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

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

- Para obtener más información sobre la API, consulta [CreateDeployment](#) la Referencia AWS SDK for Java 2.x de la API.

CreateRestApi

En el siguiente ejemplo de código, se muestra cómo usar `CreateRestApi`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

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

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

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

```
    }  
    return "";  
}
```

- Para obtener más información sobre la API, consulta [CreateRestApi](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteDeployment

En el siguiente ejemplo de código, se muestra cómo usar DeleteDeployment.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteSpecificDeployment(ApiGatewayClient apiGateway, String  
restApiId, String deploymentId) {  
  
    try {  
        DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()  
            .restApiId(restApiId)  
            .deploymentId(deploymentId)  
            .build();  
  
        apiGateway.deleteDeployment(request);  
        System.out.println("Deployment was deleted");  
  
    } catch (ApiGatewayException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obtener más información sobre la API, consulta [DeleteDeployment](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteRestApi

En el siguiente ejemplo de código, se muestra cómo usar `DeleteRestApi`.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [DeleteRestApi](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Application Auto Scaling con SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante Application Auto Scaling. AWS SDK for Java 2.x

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

DeleteScalingPolicy

En el siguiente ejemplo de código, se muestra cómo usar DeleteScalingPolicy.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteScalingPolicy](#) la Referencia AWS SDK for Java 2.x de la API.

RegisterScalableTarget

En el siguiente ejemplo de código, se muestra cómo usar RegisterScalableTarget.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [RegisterScalableTarget](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de controlador de recuperación de aplicaciones con SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante el AWS SDK for Java 2.x uso de Application Recovery Controller.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

GetRoutingControlState

En el siguiente ejemplo de código, se muestra cómo usar `GetRoutingControlState`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static GetRoutingControlStateResponse
getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn) {
    // As a best practice, we recommend choosing a random cluster endpoint to
    get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
    practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region())).build();
            return client.getRoutingControlState(
                GetRoutingControlStateRequest.builder()
                    .routingControlArn(routingControlArn).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- Para obtener más información sobre la API, consulta [GetRoutingControlState](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateRoutingControlState

En el siguiente ejemplo de código, se muestra cómo usar `UpdateRoutingControlState`.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static UpdateRoutingControlStateResponse
updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn,
    String routingControlState) {
    // As a best practice, we recommend choosing a random cluster endpoint to
    get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
    practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region()))
                .build();
            return client.updateRoutingControlState(
                UpdateRoutingControlStateRequest.builder()

.routingControlArn(routingControlArn).routingControlState(routingControlState).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- Para obtener más información sobre la API, consulta [UpdateRoutingControlState](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Aurora usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso AWS SDK for Java 2.x de Aurora.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Aurora

En el siguiente ejemplo de código se muestra cómo empezar a utilizar Aurora.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener información sobre la API, consulte [DescribeDBClusters](#) en la Referencia de la API de AWS SDK for Java 2.x .

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateDBCluster

En el siguiente ejemplo de código, se muestra cómo usar CreateDBCluster.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

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

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

- Para obtener información sobre la API, consulte [CreateDBCluster](#) en la Referencia de la API de AWS SDK for Java 2.x .

CreateDBClusterParameterGroup

En el siguiente ejemplo de código, se muestra cómo usar CreateDBClusterParameterGroup.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

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

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

- Para obtener más información sobre la API, consulta [CreateDB ClusterParameterGroup](#) en la referencia de la AWS SDK for Java 2.x API.

CreateDBClusterSnapshot

En el siguiente ejemplo de código, se muestra cómo usar CreateDBClusterSnapshot.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

```
try {
    CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
        .build();

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

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

- Para obtener más información sobre la API, consulta [CreateDB ClusterSnapshot](#) en la referencia de la AWS SDK for Java 2.x API.

CreateDBInstance

En el siguiente ejemplo de código, se muestra cómo usar CreateDBInstance.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createDBInstanceCluster(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbInstanceClusterIdentifier,
    String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
```



```

        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .engine("aurora-mysql")
        .dbInstanceClass(instanceClass)
        .build();

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

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

```

- Para obtener información sobre la API, consulte [CreateDBInstance](#) en la Referencia de la API de AWS SDK for Java 2.x .

DeleteDBCluster

En el siguiente ejemplo de código, se muestra cómo usar DeleteDBCluster.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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

```

```

        .skipFinalSnapshot(true)
        .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

```

- Para obtener información sobre la API, consulte [DeleteDBCluster](#) en la Referencia de la API de AWS SDK for Java 2.x .

DeleteDBClusterParameterGroup

En el siguiente ejemplo de código, se muestra cómo usar DeleteDBClusterParameterGroup.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();

```

```

        int listSize = instanceList.size();
        didFind = false;
        int index = 1;
        for (DBInstance instance : instanceList) {
            instanceARN = instance.dbInstanceArn();
            if (instanceARN.compareTo(clusterDBARN) == 0) {
                System.out.println(clusterDBARN + " still exists");
                didFind = true;
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.
                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }

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

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

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

```

- Para obtener más información sobre la API, consulta [DeleteDB ClusterParameterGroup en la referencia](#) de la AWS SDK for Java 2.x API.

DeleteDBInstance

En el siguiente ejemplo de código, se muestra cómo usar DeleteDBInstance.

SDK para Java 2.x

 Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

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

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

- Para obtener información sobre la API, consulte [DeleteDBInstance](#) en la Referencia de la API de AWS SDK for Java 2.x .

DescribeDBClusterParameterGroups

En el siguiente ejemplo de código, se muestra cómo usar DescribeDBClusterParameterGroups.

SDK para Java 2.x

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

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

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

- Para obtener información detallada sobre la API, consulta [DescribeDBClusterParameterGroups](#) en la referencia de la AWS SDK for Java 2.x API.

DescribeDBClusterParameters

En el siguiente ejemplo de código, se muestra cómo usar `DescribeDBClusterParameters`.

SDK para Java 2.x

 Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

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

```

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

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

```

- Para obtener información detallada sobre la API, consulta [DescribeDB ClusterParameters en la referencia de la AWS SDK for Java 2.x API](#).

DescribeDBClusterSnapshots

En el siguiente ejemplo de código, se muestra cómo usar DescribeDBClusterSnapshots.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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

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

```

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

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

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

- Para obtener información detallada sobre la API, consulta [DescribeDB ClusterSnapshots en la referencia de la AWS SDK for Java 2.x API](#).

DescribeDBClusters

En el siguiente ejemplo de código, se muestra cómo usar DescribeDBClusters.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


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

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

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

```
}
```

- Para obtener información sobre la API, consulte [DescribeDBClusters](#) en la Referencia de la API de AWS SDK for Java 2.x .

DescribeDBEngineVersions

En el siguiente ejemplo de código, se muestra cómo usar DescribeDBEngineVersions.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

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

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + 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 obtener información detallada sobre la API, consulta [DescribeDB EngineVersions en la referencia de la AWS SDK for Java 2.x API](#).

DescribeDBInstances

En el siguiente ejemplo de código, se muestra cómo usar DescribeDBInstances.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

        while (!instanceReady) {
            DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
            }
        }
    }
}
```

```
        if (instanceReadyStr.contains("available")) {
            instanceReady = true;
        } else {
            System.out.print(".");
            Thread.sleep(sleepTime * 1000);
        }
    }
}
System.out.println("Database cluster is available!");

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

- Para obtener información sobre la API, consulte [DescribeDBInstances](#) en la Referencia de la API de AWS SDK for Java 2.x .

DescribeOrderableDBInstanceOptions

En el siguiente ejemplo de código, se muestra cómo usar `DescribeOrderableDBInstanceOptions`.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

```

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

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + 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 obtener más información sobre la API, consulte la [DescribeOrderablebase de datos InstanceOptions](#) en AWS SDK for Java 2.x la referencia de API.

ModifyDBClusterParameterGroup

En el siguiente ejemplo de código, se muestra cómo usar ModifyDBClusterParameterGroup.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {

```

```
DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
    .maxRecords(20)
    .build();

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

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

- Para obtener más información sobre la API, consulta [ModifyDB ClusterParameterGroup](#) en la referencia de la AWS SDK for Java 2.x API.

Escenarios

Introducción a los clústeres de bases de datos

En el siguiente ejemplo de código, se muestra cómo:

- Cree un grupo de parámetros de clúster de base de datos de Aurora y defina los valores de los parámetros.
- Cree un clúster de base de datos que utilice el grupo de parámetros.
- Cree una instancia de base de datos que contenga una base de datos.
- Realice una instantánea del clúster de base de datos y luego limpie los recursos.

SDK para Java 2.x

 Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateDBCluster](#)
 - [Creó a B. ClusterParameterGroup](#)
 - [Creó B ClusterSnapshot](#)
 - [CreateDBInstance](#)
 - [DeleteDBCluster](#)
 - [Eliminado B ClusterParameterGroup](#)
 - [DeleteDBInstance](#)
 - [Descrito B ClusterParameterGroups](#)
 - [Descrito B ClusterParameters](#)
 - [Descrito B ClusterSnapshots](#)
 - [DescribeDBClusters](#)
 - [Descrito B EngineVersions](#)
 - [DescribeDBInstances](#)
 - [DescribeOrderableDB InstanceOptions](#)
 - [Modificar DB ClusterParameterGroup](#)

Ejemplos de escalado automático usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK for Java 2.x uso de Auto Scaling.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción al escalado automático

En los siguientes ejemplos de código se muestra cómo empezar a utilizar el escalado automático.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeAutoScalingGroups](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateAutoScalingGroup

En el siguiente ejemplo de código, se muestra cómo usar `CreateAutoScalingGroup`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateAutoScalingGroup](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteAutoScalingGroup

En el siguiente ejemplo de código, se muestra cómo usar DeleteAutoScalingGroup.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteAutoScalingGroup](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeAutoScalingGroups

En el siguiente ejemplo de código, se muestra cómo usar `DescribeAutoScalingGroups`.

SDK para Java 2.x

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeAutoScalingGroups](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeAutoScalingInstances

En el siguiente ejemplo de código, se muestra cómo usar DescribeAutoScalingInstances.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient
        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

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

- Para obtener más información sobre la API, consulta [DescribeAutoScalingInstances](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeScalingActivities

En el siguiente ejemplo de código, se muestra cómo usar `DescribeScalingActivities`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
            .maxRecords(10)
            .build();

        DescribeScalingActivitiesResponse response = autoScalingClient
            .describeScalingActivities(scalingActivitiesRequest);
        List<Activity> activities = response.activities();
        for (Activity activity : activities) {
            System.out.println("The activity Id is " + activity.activityId());
            System.out.println("The activity details are " +
activity.details());
        }


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

- Para obtener más información sobre la API, consulta [DescribeScalingActivities](#) la Referencia AWS SDK for Java 2.x de la API.

DisableMetricsCollection

En el siguiente ejemplo de código, se muestra cómo usar `DisableMetricsCollection`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
        System.out.println("The disable metrics collection operation was
successful");

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

- Para obtener más información sobre la API, consulta [DisableMetricsCollection](#) la Referencia AWS SDK for Java 2.x de la API.

EnableMetricsCollection

En el siguiente ejemplo de código, se muestra cómo usar `EnableMetricsCollection`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [EnableMetricsCollection](#) la Referencia AWS SDK for Java 2.x de la API.

SetDesiredCapacity

En el siguiente ejemplo de código, se muestra cómo usar SetDesiredCapacity.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
        .autoScalingGroupName(groupName)
        .desiredCapacity(2)
        .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

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

- Para obtener más información sobre la API, consulta [SetDesiredCapacity](#) la Referencia AWS SDK for Java 2.x de la API.

TerminateInstanceInAutoScalingGroup

En el siguiente ejemplo de código, se muestra cómo usar `TerminateInstanceInAutoScalingGroup`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
        TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

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

- Para obtener más información sobre la API, consulta [TerminateInstanceInAutoScalingGroup](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateAutoScalingGroup

En el siguiente ejemplo de código, se muestra cómo usar UpdateAutoScalingGroup.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
    String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
            .waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("You successfully updated the auto scaling group " +
groupName);

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

- Para obtener más información sobre la API, consulta [UpdateAutoScalingGroup](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Creación y administración de un servicio resiliente

El siguiente ejemplo de código muestra cómo crear un servicio web con equilibrio de carga que muestre recomendaciones de libros, películas y canciones. El ejemplo muestra cómo responde el servicio a los errores y cómo reestructurarlo para aumentar la resiliencia cuando se produzcan errores.

- Utilice un grupo de Amazon EC2 Auto Scaling para crear instancias de Amazon Elastic Compute Cloud (Amazon EC2) basadas en una plantilla de lanzamiento y para mantener el número de instancias dentro de un rango específico.
- Administre y distribuya las solicitudes HTTP con Elastic Load Balancing.
- Supervise el estado de las instancias de un grupo de escalado automático y reenvíe las solicitudes solo a las instancias en buen estado.
- Ejecute un servidor web Python en cada instancia de EC2 para administrar las solicitudes HTTP. El servidor web responde con recomendaciones y comprobaciones de estado.
- Simule un servicio de recomendaciones con una tabla de Amazon DynamoDB.
- Controle la respuesta del servidor web a las solicitudes y las comprobaciones de estado actualizando AWS Systems Manager los parámetros.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute el escenario interactivo en un símbolo del sistema.

```
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);
}

```

```
}  
}
```

Cree una clase que agrupe las acciones de escalado automático y 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.");
    }
}

```

Cree una clase que resuma las acciones de 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 "";
}
}

```

Cree una clase que utilice DynamoDB para simular un servicio de recomendaciones.

```

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);
}
}

```

Cree una clase que agrupe las acciones de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de 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](#)

Administrar grupos e instancias

En el siguiente ejemplo de código, se muestra cómo:

- Crear un grupo de Amazon EC2 Auto Scaling con una plantilla de lanzamiento y zonas de disponibilidad y obtener información sobre las instancias en ejecución
- Habilita la recopilación de CloudWatch métricas de Amazon.
- Actualizar la capacidad deseada del grupo y esperar a que una instancia se inicie
- Terminar una instancia del grupo.
- Mostrar las actividades de escalado que se producen como respuesta a las solicitudes de los usuarios y a los cambios de capacidad
- Obtén estadísticas para CloudWatch las métricas y, a continuación, limpia los recursos.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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,
groupName);
if (instanceId.compareTo("") == 0) {
```

```
        System.out.println("Error - no instance Id value");
        System.exit(1);
    } else {
        System.out.println("The instance Id value is " + instanceId);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
    describeAutoScalingInstance(autoScalingClient, instanceId);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Enable metrics collection " + instanceId);
    enableMetricsCollection(autoScalingClient, groupName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Update an Auto Scaling group to update max size to
3");
    updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Describe Auto Scaling groups");
    describeAutoScalingGroups(autoScalingClient, groupName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Describe account details");
    describeAccountLimits(autoScalingClient);
    System.out.println(
        "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
    Thread.sleep(60000);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("8. Set desired capacity to 2");
    setDesiredCapacity(autoScalingClient, groupName);
    System.out.println(DASHES);

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

```
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for
the group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

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

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                .autoScalingGroupName(groupName)
                .maxRecords(10)
                .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
```

```
        for (Activity activity : activities) {
            System.out.println("The activity Id is " + activity.activityId());
            System.out.println("The activity details are " +
activity.details());
        }

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

public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
            .autoScalingGroupName(groupName)
            .desiredCapacity(2)
            .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

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

public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
String launchTemplateName,
String vpcZoneId) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
```

```
        .availabilityZones("us-east-1a")
        .launchTemplate(templateSpecification)
        .maxSize(1)
        .minSize(1)
        .vpcZoneIdentifier(vpcZoneId)
        .build();

    autoScalingClient.createAutoScalingGroup(request);
    DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupExists(groupsRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Auto Scaling Group created");

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

public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
            .builder()
            .instanceIds(id)
            .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient

        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }
    }
}
```



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

public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .maxRecords(10)
            .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("*** The service to use for the health checks: "
+ group.healthCheckType());
        }

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

public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response = autoScalingClient
            .describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
        }
    }
}
```

```
        System.out.println("The group ARN is " +
group.autoScalingGroupARN());
        List<Instance> instances = group.instances();

        for (Instance instance : instances) {
            instanceId = instance.instanceId();
            System.out.println("The instance id is " + instanceId);
            System.out.println("The lifecycle state is " +
instance.lifecycleState());
        }
    }

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

public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

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

public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
```

```
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
    .autoScalingGroupName(groupName)
    .metrics("GroupMaxSize")
    .build();

autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
    System.out.println("The disable metrics collection operation was
successful");

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

public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
    try {
        DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
        System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
        System.out.println("The current number of auto scaling groups is " +
response.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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Ejemplos de Amazon Bedrock usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Amazon Bedrock.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

GetFoundationModel

En el siguiente ejemplo de código, se muestra cómo usar `GetFoundationModel`.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtenga detalles sobre un modelo básico mediante el cliente sincrónico de 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);
}
}

```

Obtenga detalles sobre un modelo básico mediante el cliente asíncrono 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 obtener más información sobre la API, consulte [GetFoundationModel](#) la referencia de la API. AWS SDK for Java 2.x

ListFoundationModels

En el siguiente ejemplo de código, se muestra cómo usar `ListFoundationModels`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Enumere los modelos de base de Amazon Bedrock disponibles mediante el cliente sincrónico de 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);
    }
}

```

Enumere los modelos de base de Amazon Bedrock disponibles mediante el cliente asíncrono de 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 obtener más información sobre la API, consulte la referencia de la API.

[ListFoundationModels](#) AWS SDK for Java 2.x

Ejemplos de Amazon Bedrock Runtime usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante Amazon Bedrock Runtime. AWS SDK for Java 2.x

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Jurassic-2 de AI21 Labs](#)
- [Amazon Titan Image Generator](#)
- [Texto de Amazon Titan](#)

- [Incrustaciones de texto de Amazon Titan](#)
- [Anthropic Claude](#)
- [Cohere Command](#)
- [Meta Llama](#)
- [API Mistral](#)
- [Escenarios](#)
- [Difusión estable](#)

Jurassic-2 de AI21 Labs

Conversar

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a AI21 Labs Jurassic-2 mediante la API Converse de Bedrock.

SDK para Java 2.x

Note

Hay más información. [GitHub](#) Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envía un mensaje de texto a AI21 Labs Jurassic-2 mediante la API Converse de Bedrock.

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {
```

```
// Create a Bedrock Runtime client in the AWS Region you want to use.
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Jurassic-2 Mid.
var modelId = "ai21.j2-mid-v1";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

try {
    // Send the message with a basic inference configuration.
    ConverseResponse response = client.converse(request -> request
        .modelId(modelId)
        .messages(message)
        .inferenceConfig(config -> config
            .maxTokens(512)
            .temperature(0.5F)
            .topP(0.9F)));

    // Retrieve the generated text from Bedrock's response object.
    var responseText = response.output().message().content().get(0).text();
    System.out.println(responseText);

    return responseText;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}
```

```
    public static void main(String[] args) {
        converse();
    }
}
```

Envía un mensaje de texto a AI21 Labs Jurassic-2 utilizando la API Converse de Bedrock con el cliente Java asíncrono.

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
```

```
        .role(ConversationRole.USER)
        .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
    System.out.println(responseText);

    return responseText;

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    converseAsync();
}
```

```
}  
}
```

- [Para obtener más información sobre la API, consulte Converse en la referencia de la API.AWS SDK for Java 2.x](#)

InvokeModel

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a AI21 Labs Jurassic-2 mediante la API Invoke Model.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto.

```
// Use the native inference API to send a text message to AI21 Labs Jurassic-2.  
  
import org.json.JSONObject;  
import org.json.JSONPointer;  
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;  
import software.amazon.awssdk.core.SdkBytes;  
import software.amazon.awssdk.core.exception.SdkClientException;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;  
  
public class InvokeModel {  
  
    public static String invokeModel() {  
  
        // Create a Bedrock Runtime client in the AWS Region you want to use.  
        // Replace the DefaultCredentialsProvider with your preferred credentials  
        provider.  
        var client = BedrockRuntimeClient.builder()  
            .credentialsProvider(DefaultCredentialsProvider.create())  
            .region(Region.US_EAST_1)
```



```
        .build();

// Set the model ID, e.g., Jurassic-2 Mid.
var modelId = "ai21.j2-mid-v1";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
jurassic2.html
var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/completions/0/data/
text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    invokeModel();
}
```

```
}  
}
```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia de AWS SDK for Java 2.x la API.

Amazon Titan Image Generator

InvokeModel

El siguiente ejemplo de código muestra cómo invocar Amazon Titan Image en Amazon Bedrock para generar una imagen.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree una imagen con el generador de imágenes Amazon Titan.

```
// Create an image with the Amazon Titan Image Generator.  
  
import org.json.JSONObject;  
import org.json.JSONPointer;  
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;  
import software.amazon.awssdk.core.SdkBytes;  
import software.amazon.awssdk.core.exception.SdkClientException;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;  
  
import java.math.BigInteger;  
import java.security.SecureRandom;  
  
import static com.example.bedrockruntime.libs.ImageTools.displayImage;  
  
public class InvokeModel {  
  
    public static String invokeModel() {
```

```
// Create a Bedrock Runtime client in the AWS Region you want to use.
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Titan Image G1.
var modelId = "amazon.titan-image-generator-v1";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html
var nativeRequestTemplate = """
    {
        "taskType": "TEXT_IMAGE",
        "textToImageParams": { "text": "{{prompt}}" },
        "imageGenerationConfig": { "seed": {{seed}} }
    }""";

// Define the prompt for the image generation.
var prompt = "A stylized picture of a cute old steampunk robot";

// Get a random 31-bit seed for the image generation (max. 2,147,483,647).
var seed = new BigInteger(31, new SecureRandom());

// Embed the prompt and seed in the model's native request payload.
var nativeRequest = nativeRequestTemplate
    .replace("{{prompt}}", prompt)
    .replace("{{seed}}", seed.toString());

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());
```

```
        // Retrieve the generated image data from the model's response.
        var base64ImageData = new JSONPointer("/
images/0").queryFrom(responseBody).toString();

        return base64ImageData;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    System.out.println("Generating image. This may take a few seconds...");

    String base64ImageData = invokeModel();

    displayImage(base64ImageData);
}
}
```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia AWS SDK for Java 2.x de la API.

Texto de Amazon Titan

Conversar

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Amazon Titan Text mediante la API Converse de Bedrock.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envía un mensaje de texto a Amazon Titan Text mediante la API Converse de Bedrock.

```
// Use the Converse API to send a text message to Amazon Titan Text.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
```

```

        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F));

    // Retrieve the generated text from Bedrock's response object.
    var responseText = response.output().message().content().get(0).text();
    System.out.println(responseText);

    return responseText;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    converse();
}
}

```

Envíe un mensaje de texto a Amazon Titan Text mediante la API Converse de Bedrock con el cliente Java asíncrono.

```

// Use the Converse API to send a text message to Amazon Titan Text
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

```

```
// Create a Bedrock Runtime client in the AWS Region you want to use.
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeAsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Titan Text Premier.
var modelId = "amazon.titan-text-premier-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
}
```

```
    });

    try {
        // Wait for the future object to complete and retrieve the generated
text.
        String responseText = future.get();
        System.out.println(responseText);

        return responseText;
    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- Para obtener más información sobre la API, consulte [Converse](#) en la referencia de la API.AWS SDK for Java 2.x

ConverseStream

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Amazon Titan Text mediante la API Converse de Bedrock y cómo procesar el flujo de respuestas en tiempo real.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envíe un mensaje de texto a Amazon Titan Text mediante la API Converse de Bedrock y procese el flujo de respuestas en tiempo real.


```
// Use the Converse API to send a text message to Amazon Titan Text
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
```

```

        System.out.print(responseText);
    }).build()
    ).onError(err ->
        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
    ).build();

    try {
        // Send the message with a basic inference configuration and attach the
handler.
        client.converseStream(request -> request
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
}

```

- Para obtener más información sobre la API, consulte la Referencia de [ConverseStream](#) la AWS SDK for Java 2.x API.

InvokeModel

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Amazon Titan Text mediante la API Invoke Model.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto.

```
// Use the native inference API to send a text message to Amazon Titan Text.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        titan-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                .body(SdkBytes.fromUtf8String(nativeRequest))
```

```
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/results/0/outputText").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia de AWS SDK for Java 2.x la API.

InvokeModelWithResponseStream

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a los modelos de Amazon Titan Text mediante la API Invoke Model e imprimir el flujo de respuestas.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto y procesar el flujo de respuestas en tiempo real.

```
// Use the native inference API to send a text message to Amazon Titan Text
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        titan-text.html
```

```
var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/outputText").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
    client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

    // Return the complete response text.
    return completeResponseTextBuffer.toString();

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    throw new RuntimeException(e);
}
```

```
    }  
  }  
  
  public static void main(String[] args) throws ExecutionException,  
    InterruptedException {  
    invokeModelWithResponseStream();  
  }  
}
```

- Para obtener más información sobre la API, consulte [InvokeModelWithResponseStream](#) la referencia de AWS SDK for Java 2.x la API.

Incrustaciones de texto de Amazon Titan

InvokeModel

En el siguiente ejemplo de código, se muestra cómo:

- Comience a crear su primera incrustación.
- Cree incrustaciones configurando el número de dimensiones y la normalización (solo en la versión 2).

SDK para Java 2.x

Note

Hay más en marcha. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crea tu primera incrustación con Titan Text Embeddings V2.

```
// Generate and print an embedding with Amazon Titan Text Embeddings.  
  
import org.json.JSONObject;  
import org.json.JSONPointer;  
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;  
import software.amazon.awssdk.core.SdkBytes;  
import software.amazon.awssdk.core.exception.SdkClientException;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Embeddings V2.
        var modelId = "amazon.titan-embed-text-v2:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        titan-embed-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{inputText}}\" }";

        // The text to convert into an embedding.
        var inputText = "Please recommend books with a theme similar to the movie
        'Inception.'";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{inputText}}",
        inputText);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
```



```

        var text = new JSONPointer("/
embedding").queryFrom(responseBody).toString();
        System.out.println(text);

        return text;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    invokeModel();
}
}

```

Invoca Titan Text Embeddings V2 configurando el número de dimensiones y la normalización.

```

/**
 * Invoke Amazon Titan Text Embeddings V2 with additional inference parameters.
 *
 * @param inputText - The text to convert to an embedding.
 * @param dimensions - The number of dimensions the output embeddings should
have.
 *
 *           Values accepted by the model: 256, 512, 1024.
 * @param normalize - A flag indicating whether or not to normalize the output
embeddings.
 * @return The {@link JSONObject} representing the model's response.
 */
public static JSONObject invokeModel(String inputText, int dimensions, boolean
normalize) {

    // Create a Bedrock Runtime client in the AWS Region of your choice.
    var client = BedrockRuntimeClient.builder()
        .region(Region.US_WEST_2)
        .build();

    // Set the model ID, e.g., Titan Embed Text v2.0.
    var modelId = "amazon.titan-embed-text-v2:0";

```

```
// Create the request for the model.
var nativeRequest = """
    {
        "inputText": "%s",
        "dimensions": %d,
        "normalize": %b
    }
    """.formatted(inputText, dimensions, normalize);

// Encode and send the request.
var response = client.invokeModel(request -> {
    request.body(SdkBytes.fromUtf8String(nativeRequest));
    request.modelId(modelId);
});

// Decode the model's response.
var modelResponse = new JSONObject(response.body().asUtf8String());

// Extract and print the generated embedding and the input text token count.
var embedding = modelResponse.getJSONArray("embedding");
var inputTokenCount = modelResponse.getBigInteger("inputTextTokenCount");
System.out.println("Embedding: " + embedding);
System.out.println("\nInput token count: " + inputTokenCount);

// Return the model's native response.
return modelResponse;
}
```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia de la API.AWS SDK for Java 2.x

Anthropic Claude

Conversar

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Anthropic Claude mediante la API Converse de Bedrock.

SDK para Java 2.x

 Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envía un mensaje de texto a Anthropic Claude utilizando la API Converse de Bedrock.

```
// Use the Converse API to send a text message to Anthropic Claude.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
```

```
        .build());

    try {
        // Send the message with a basic inference configuration.
        ConverseResponse response = client.converse(request -> request
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)));

        // Retrieve the generated text from Bedrock's response object.
        var responseText = response.output().message().content().get(0).text();
        System.out.println(responseText);

        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
            e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converse();
}
}
```

Envía un mensaje de texto a Anthropic Claude utilizando la API Converse de Bedrock con el cliente Java asíncrono.

```
// Use the Converse API to send a text message to Anthropic Claude
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
```

```
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
```

```
        if (error == null) {
            // Extract the generated text from Bedrock's response object.
            String responseText =
response.output().message().content().get(0).text();
            future.complete(responseText);
        } else {
            future.completeExceptionally(error);
        }
    });

    try {
        // Wait for the future object to complete and retrieve the generated
text.
        String responseText = future.get();
        System.out.println(responseText);

        return responseText;
    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- [Para obtener más información sobre la API, consulte Converse en la referencia de la API.AWS SDK for Java 2.x](#)

ConverseStream

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Anthropic Claude mediante la API Converse de Bedrock y procesar el flujo de respuestas en tiempo real.

SDK para Java 2.x

 Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envía un mensaje de texto a Anthropic Claude mediante la API Converse de Bedrock y procesa el flujo de respuestas en tiempo real.

```
// Use the Converse API to send a text message to Anthropic Claude
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
```

```
    var inputText = "Describe the purpose of a 'hello world' program in one
line.";
    var message = Message.builder()
        .content(ContentBlock.fromText(inputText))
        .role(ConversationRole.USER)
        .build();

    // Create a handler to extract and print the response text in real-time.
    var responseStreamHandler = ConverseStreamResponseHandler.builder()
        .subscriber(ConverseStreamResponseHandler.Visitor.builder()
            .onContentBlockDelta(chunk -> {
                String responseText = chunk.delta().text();
                System.out.print(responseText);
            }).build()
        ).onError(err ->
            System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
        ).build();

    try {
        // Send the message with a basic inference configuration and attach the
handler.
        client.converseStream(request -> request.modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
```

- Para obtener más información sobre la API, consulte la referencia de [ConverseStream](#) la AWS SDK for Java 2.x API.

InvokeModel

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Anthropic Claude mediante la API Invoke Model.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto.

```
// Use the native inference API to send a text message to Anthropic Claude.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
```

```
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
anthropic-claude-messages.html
var nativeRequestTemplate = """
    {
        "anthropic_version": "bedrock-2023-05-31",
        "max_tokens": 512,
        "temperature": 0.5,
        "messages": [{
            "role": "user",
            "content": "{{prompt}}"
        }]
    }""";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/content/0/
text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
```

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

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia de AWS SDK for Java 2.x la API.

InvokeModelWithResponseStream

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a los modelos Anthropic Claude mediante la API Invoke Model e imprimir el flujo de respuestas.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto y procesar el flujo de respuestas en tiempo real.

```
// Use the native inference API to send a text message to Anthropic Claude
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

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

```
import static
software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
    InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
    provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
    at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
    anthropic-claude-messages.html
        var nativeRequestTemplate = """
            {
                "anthropic_version": "bedrock-2023-05-31",
                "max_tokens": 512,
                "temperature": 0.5,
                "messages": [{
                    "role": "user",
                    "content": "{{prompt}}"
                }]
            }""";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        // Create a request with the model ID and the model's native request
    payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
```

```
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
        .build();

    // Prepare a buffer to accumulate the generated response text.
    var completeResponseTextBuffer = new StringBuilder();

    // Prepare a handler to extract, accumulate, and print the response text in
    real-time.
    var responseStreamHandler =
    InvokeModelWithResponseStreamResponseHandler.builder()
        .subscriber(Visitor.builder().onChunk(chunk -> {
            var response = new JSONObject(chunk.bytes().asUtf8String());

            // Extract and print the text from the content blocks.
            if (Objects.equals(response.getString("type"),
"content_block_delta")) {
                var text = new JSONPointer("/delta/
text").queryFrom(response);
                System.out.print(text);

                // Append the text to the response text buffer.
                completeResponseTextBuffer.append(text);
            }
        })).build()).build();

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
```

```
}  
}
```

- Para obtener más información sobre la API, consulte [InvokeModelWithResponseStream](#) la referencia de AWS SDK for Java 2.x la API.

Cohere Command

Conversar

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Cohere Command mediante la API Converse de Bedrock.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envía un mensaje de texto a Cohere Command mediante la API Converse de Bedrock.

```
// Use the Converse API to send a text message to Cohere Command.  
  
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;  
import software.amazon.awssdk.core.exception.SdkClientException;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;  
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;  
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;  
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;  
import software.amazon.awssdk.services.bedrockruntime.model.Message;  
  
public class Converse {  
  
    public static String converse() {  
  
        // Create a Bedrock Runtime client in the AWS Region you want to use.  
        // Replace the DefaultCredentialsProvider with your preferred credentials  
        provider.
```

```
var client = BedrockRuntimeClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

try {
    // Send the message with a basic inference configuration.
    ConverseResponse response = client.converse(request -> request
        .modelId(modelId)
        .messages(message)
        .inferenceConfig(config -> config
            .maxTokens(512)
            .temperature(0.5F)
            .topP(0.9F)));

    // Retrieve the generated text from Bedrock's response object.
    var responseText = response.output().message().content().get(0).text();
    System.out.println(responseText);

    return responseText;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    converse();
}
```

```
}
```

Envía un mensaje de texto a Cohere Command mediante la API Converse de Bedrock con el cliente Java asíncrono.

```
// Use the Converse API to send a text message to Cohere Command
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();
```



```
// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
    System.out.println(responseText);

    return responseText;
} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
    throw new RuntimeException(e);
}
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- [Para obtener más información sobre la API, consulte Converse en la referencia de la API.AWS SDK for Java 2.x](#)

ConverseStream

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Cohere Command mediante la API Converse de Bedrock y cómo procesar el flujo de respuestas en tiempo real.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envía un mensaje de texto a Cohere Command mediante la API Converse de Bedrock y procesa el flujo de respuestas en tiempo real.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeAsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Create a handler to extract and print the response text in real-time.
var responseStreamHandler = ConverseStreamResponseHandler.builder()
    .subscriber(ConverseStreamResponseHandler.Visitor.builder()
        .onContentBlockDelta(chunk -> {
            String responseText = chunk.delta().text();
            System.out.print(responseText);
        }).build()
    ).onError(err ->
        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
    ).build();

try {
    // Send the message with a basic inference configuration and attach the
handler.
    client.converseStream(request -> request.modelId(modelId)
        .messages(message)
        .inferenceConfig(config -> config
            .maxTokens(512)
            .temperature(0.5F)
            .topP(0.9F)
        ), responseStreamHandler).get();

} catch (ExecutionException | InterruptedException e) {
```

```
        System.err.printf("Can't invoke '%s': %s", modelId,
        e.getCause().getMessage());
    }
}
}
```

- Para obtener más información sobre la API, consulte la referencia de [ConverseStream](#) la AWS SDK for Java 2.x API.

InvokeModel: Comando R y R+

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Cohere Command R y R+ mediante la API Invoke Model.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto.

```
// Use the native inference API to send a text message to Cohere Command R.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Command_R_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
```

```
var client = BedrockRuntimeClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command-r-plus.html
var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}
```

```
    public static void main(String[] args) {
        invokeModel();
    }
}
```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia de AWS SDK for Java 2.x la API.

InvokeModel: Comando y Comando Light

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Cohere Command mediante la API Invoke Model.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto.

```
// Use the native inference API to send a text message to Cohere Command.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Command_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
```

```
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_EAST_1)
        .build();

// Set the model ID, e.g., Command Light.
var modelId = "cohere.command-light-text-v14";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command.html
var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/generations/0/
text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}
```

```
public static void main(String[] args) {
    invokeModel();
}
}
```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia de AWS SDK for Java 2.x la API.

InvokeModelWithResponseStream: Comando R y R+

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Cohere Command mediante la API Invoke Model con un flujo de respuesta.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto y procesar el flujo de respuestas en tiempo real.

```
// Use the native inference API to send a text message to Cohere Command R
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

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



```
import static
software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class Command_R_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
    InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command-r-plus.html
        var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        // Create a request with the model ID and the model's native request
payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
            .body(SdkBytes.fromUtf8String(nativeRequest))
            .modelId(modelId)
            .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
real-time.
```

```

    var responseStreamHandler =
    InvokeModelWithResponseStreamResponseHandler.builder()
        .subscriber(Visitor.builder().onChunk(chunk -> {
            // Extract and print the text from the model's native response.
            var response = new JSONObject(chunk.bytes().asUtf8String());
            var text = new JSONPointer("/text").queryFrom(response);
            System.out.print(text);

            // Append the text to the response text buffer.
            completeResponseTextBuffer.append(text);
        })).build()).build();

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
        responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
        e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}

```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia de AWS SDK for Java 2.x la API.

InvokeModelWithResponseStream: Comando y Comando Light

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Cohere Command mediante la API Invoke Model con un flujo de respuesta.

SDK para Java 2.x

 Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto y procesar el flujo de respuestas en tiempo real.

```
// Use the native inference API to send a text message to Cohere Command
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Command_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Command Light.
var modelId = "cohere.command-light-text-v14";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command.html
var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generations/0/
text").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
```

```
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia de AWS SDK for Java 2.x la API.

Meta Llama

Conversar

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Meta Llama mediante la API Converse de Bedrock.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envía un mensaje de texto a Meta Llama utilizando la API Converse de Bedrock.

```
// Use the Converse API to send a text message to Meta Llama.
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));
        }
    }
}
```

```
        // Retrieve the generated text from Bedrock's response object.
        var responseText = response.output().message().content().get(0).text();
        System.out.println(responseText);

        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converse();
}
}
```

Envía un mensaje de texto a Meta Llama utilizando la API Converse de Bedrock con el cliente Java asíncrono.

```
// Use the Converse API to send a text message to Meta Llama
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
```

```
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_EAST_1)
        .build();

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
```



```
        String responseText = future.get();
        System.out.println(responseText);

        return responseText;

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- Para obtener más información sobre la API, consulta [Converse](#) en la referencia de la API.AWS SDK for Java 2.x

ConverseStream

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Meta Llama mediante la API Converse de Bedrock y cómo procesar el flujo de respuestas en tiempo real.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envía un mensaje de texto a Meta Llama utilizando la API Converse de Bedrock y procesa el flujo de respuestas en tiempo real.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
                System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
            ).build();
```

```
    try {
        // Send the message with a basic inference configuration and attach the
handler.
        client.converseStream(request -> request
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
```

- Para obtener más información sobre la API, consulta la Referencia de [ConverseStream](#) la AWS SDK for Java 2.x API.

InvokeModel: Llama 2

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Meta Llama 2 mediante la API Invoke Model.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto.

```
// Use the native inference API to send a text message to Meta Llama 2.

import org.json.JSONObject;
```

```
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama2_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 2 Chat 13B.
        var modelId = "meta.llama2-13b-chat-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\n\"}";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 2's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\n".replace("{{prompt}}",
        prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
        instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                .body(SdkBytes.fromUtf8String(nativeRequest))
```

```
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia de AWS SDK for Java 2.x la API.

InvokeModel: Llama 3

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Meta Llama 3 mediante la API Invoke Model.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto.

```
// Use the native inference API to send a text message to Meta Llama 3.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama3_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 3's instruction format.
        var instruction = (
            "<|begin_of_text|>\n" +
            "<|start_header_id|>user<|end_header_id|>\n" +
            "{{prompt}} <|eot_id|>\n" +
            "<|start_header_id|>assistant<|end_header_id|>\n"
        ).replace("{{prompt}}", prompt);
```

```
// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia de AWS SDK for Java 2.x la API.

InvokeModelWithResponseStream: Llama 2

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Meta Llama 2, mediante la API Invoke Model, e imprimir el flujo de respuesta.

SDK para Java 2.x

 Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto y procesar el flujo de respuestas en tiempo real.

```
// Use the native inference API to send a text message to Meta Llama 2
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Llama2_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```



```
// Set the model ID, e.g., Llama 2 Chat 13B.
var modelId = "meta.llama2-13b-chat-v1";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Llama 2's instruction format.
var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generation").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    })
    )
    .build();
```

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

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Para obtener más información sobre la API, consulte [InvokeModelWithResponseStream](#) la referencia de AWS SDK for Java 2.x la API.

InvokeModelWithResponseStream: Llama 3

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Meta Llama 3, utilizando la API Invoke Model, e imprimir el flujo de respuesta.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API `Invoke Model` para enviar un mensaje de texto y procesar el flujo de respuestas en tiempo real.

```
// Use the native inference API to send a text message to Meta Llama 3
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Llama3_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
```

```
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Llama 3's instruction format.
var instruction = (
    "<|begin_of_text|>\n" +
    "<|start_header_id|>user<|end_header_id|>\n" +
    "{{prompt}} <|eot_id|>\n" +
    "<|start_header_id|>assistant<|end_header_id|>\n"
).replace("{{prompt}}", prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONObject("/generation").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
```

```
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Para obtener más información sobre la API, consulte [InvokeModelWithResponseStream](#) la referencia de AWS SDK for Java 2.x la API.

API Mistral

Conversar

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Mistral mediante la API Converse de Bedrock.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envía un mensaje de texto a Mistral utilizando la API Converse de Bedrock.

```
// Use the Converse API to send a text message to Mistral.
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));
        }
    }
}
```

```
        // Retrieve the generated text from Bedrock's response object.
        var responseText = response.output().message().content().get(0).text();
        System.out.println(responseText);

        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }

}

public static void main(String[] args) {
    converse();
}
}
```

Envía un mensaje de texto a Mistral utilizando la API Converse de Bedrock con el cliente Java asíncrono.

```
// Use the Converse API to send a text message to Mistral
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
```

```
var client = BedrockRuntimeAsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
```



```
        // Wait for the future object to complete and retrieve the generated
text.
        String responseText = future.get();
        System.out.println(responseText);

        return responseText;

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- [Para obtener más información sobre la API, consulta Converse en la referencia de la API.AWS SDK for Java 2.x](#)

ConverseStream

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Mistral mediante la API Converse de Bedrock y cómo procesar el flujo de respuestas en tiempo real.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envía un mensaje de texto a Mistral mediante la API Converse de Bedrock y procesa el flujo de respuestas en tiempo real.

```
// Use the Converse API to send a text message to Mistral
// and print the response stream.
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
```

```
        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
        ).build();

    try {
        // Send the message with a basic inference configuration and attach the
handler.
        client.converseStream(request -> request.modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
```

- Para obtener más información sobre la API, consulte [ConverseStream](#) la Referencia de la API. AWS SDK for Java 2.x

InvokeModel

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a los modelos Mistral mediante la API Invoke Model.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto.

```
// Use the native inference API to send a text message to Mistral.
```

```
import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        mistral-text-completion.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Mistral's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
        prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
        instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
```

```
var response = client.invokeModel(request -> request
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
);

// Decode the response body.
var responseBody = new JSONObject(response.body().asUtf8String());

// Retrieve the generated text from the model's response.
var text = new JSONPointer("/outputs/0/
text").queryFrom(responseBody).toString();
System.out.println(text);

return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia de AWS SDK for Java 2.x la API.

InvokeModelWithResponseStream

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a los modelos de IA de Mistral, mediante la API Invoke Model, e imprimir el flujo de respuesta.

SDK para Java 2.x

 Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la API Invoke Model para enviar un mensaje de texto y procesar el flujo de respuestas en tiempo real.

```
// Use the native inference API to send a text message to Mistral
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
mistral-text-completion.html
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Mistral's instruction format.
var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/outputs/0/
text").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
```

```
        completeResponseTextBuffer.append(text);
    }).build()).build();

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Para obtener más información sobre la API, consulte [InvokeModelWithResponseStream](#) la referencia de AWS SDK for Java 2.x la API.

Escenarios

Crear una aplicación de sitio de pruebas que interactúe con modelos fundacionales de Amazon Bedrock

En el siguiente ejemplo de código se muestra cómo crear sitios de pruebas que interactúan con modelos fundacionales de Amazon Bedrock a través de diferentes modalidades.

SDK para Java 2.x

El sitio de pruebas del modelo fundacional (FM) de Java es una aplicación de muestra de Spring Boot que muestra cómo utilizar Amazon Bedrock con Java. En este ejemplo se muestra cómo los desarrolladores de Java pueden utilizar Amazon Bedrock para crear aplicaciones habilitadas para

IA generativa. Puede probar los modelos fundacionales de Amazon Bedrock e interactuar con ellos mediante los tres sitios de pruebas siguientes:

- Un sitio de pruebas de texto.
- Un sitio de pruebas de chat.
- Un sitio de pruebas de imágenes.

En el ejemplo también se enumeran y muestran los modelos fundacionales a los que tiene acceso y sus características. Para ver el código fuente y las instrucciones de implementación, consulta el proyecto en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Bedrock Runtime

Difusión estable

InvokeModel

El siguiente ejemplo de código muestra cómo invocar Stability.ai Stable Diffusion XL en Amazon Bedrock para generar una imagen.

SDK para Java 2.x

Note

Hay más información al respecto. [Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.](#)

Crea una imagen con Stable Diffusion.

```
// Create an image with Stable Diffusion.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
```

```
import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Stable Diffusion XL v1.
        var modelId = "stability.stable-diffusion-xl-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        diffusion-1-0-text-image.html
        var nativeRequestTemplate = ""
            {
                "text_prompts": [{ "text": "{{prompt}}" }],
                "style_preset": "{{style}}",
                "seed": {{seed}}
            }"";

        // Define the prompt for the image generation.
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random 32-bit seed for the image generation (max. 4,294,967,295).
        var seed = new BigInteger(31, new SecureRandom());

        // Choose a style preset.
        var style = "cinematic";

        // Embed the prompt, seed, and style in the model's native request payload.
        String nativeRequest = nativeRequestTemplate
            .replace("{{prompt}}", prompt)
```

```
        .replace("{{seed}}", seed.toString())
        .replace("{{style}}", style);

    try {
        // Encode and send the request to the Bedrock Runtime.
        var response = client.invokeModel(request -> request
            .body(SdkBytes.fromUtf8String(nativeRequest))
            .modelId(modelId)
        );

        // Decode the response body.
        var responseBody = new JSONObject(response.body().asUtf8String());

        // Retrieve the generated image data from the model's response.
        var base64ImageData = new JSONPointer("/artifacts/0/base64")
            .queryFrom(responseBody)
            .toString();

        return base64ImageData;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
            e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    System.out.println("Generating image. This may take a few seconds...");

    String base64ImageData = invokeModel();

    displayImage(base64ImageData);
}
}
```

- Para obtener más información sobre la API, consulte [InvokeModel](#) la referencia AWS SDK for Java 2.x de la API.

CloudFront ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with CloudFront.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateDistribution

En el siguiente ejemplo de código, se muestra cómo usar `CreateDistribution`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

El siguiente ejemplo utiliza un bucket de Amazon Simple Storage Service (Amazon S3) como un origen de contenido.

Tras crear la distribución, el código crea un modo de esperar [CloudFrontWaiter](#) que se despliegue la distribución antes de devolverla.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

    private static final Logger logger =
        LoggerFactory.getLogger(CreateDistribution.class);

    public static Distribution createDistribution(CloudFrontClient
        cloudFrontClient, S3Client s3Client,
        final String bucketName, final String keyGroupId, final
        String originAccessControlId) {

        final String region = s3Client.headBucket(b ->
        b.bucket(bucketName)).sdkHttpResponse().headers()
            .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region +
        ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for
        the originId.

        // The service API requires some deprecated methods, such as
        // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
        CreateDistributionResponse createDistResponse =
        cloudFrontClient.createDistribution(builder -> builder
            .distributionConfig(b1 -> b1
                .origins(b2 -> b2
                    .quantity(1)
                    .items(b3 -> b3

                .domainName(originDomain)

```

```
.id(originId)

.s3OriginConfig(builder4 -> builder4
    .originAccessIdentity(
        ""))

.originAccessControlId(
    originAccessControlId)))

    .defaultCacheBehavior(b2 -> b2

.viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

    .minTTL(200L)
    .forwardedValues(b5
-> b5

.cookies(cp -> cp
    .forward(ItemSelection.NONE))

.queryString(true))

    .trustedKeyGroups(b3
-> b3

.quantity(1)

.items(keyGroupId)

.enabled(true))

    .allowedMethods(b4 -
> b4

.quantity(2)

.items(Method.HEAD, Method.GET)

.cachedMethods(b5 -> b5

    .quantity(2)
```

```
        .items(Method.HEAD,
                Method.GET))))
        .cacheBehaviors(b -> b
            .quantity(1)
            .items(b2 -> b2

.pathPattern("/index.html")

.viewerProtocolPolicy(
    ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3
    .quantity(1)
    .items(keyGroupId)
    .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4
    .cookies(cp -> cp
        .forward(ItemSelection.NONE))
    .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)
    .items(Method.HEAD,
            Method.GET)
    .cachedMethods(b6 -> b6
        .quantity(2)
```

```

        .items(Method.HEAD,
                Method.GET))))))
            .enabled(true)
            .comment("Distribution built with
java")

        .callerReference(Instant.now().toString()));

        final Distribution distribution = createDistResponse.distribution();
        logger.info("Distribution created. DomainName: [{}] Id: [{}]",
distribution.domainName(),
                distribution.id());
        logger.info("Waiting for distribution to be deployed ...");
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
                .waitUntilDistributionDeployed(builder ->
builder.id(distribution.id()))
                .matched();
            responseOrException.response()
                .orElseThrow(() -> new
RuntimeException("Distribution not created"));
            logger.info("Distribution deployed. DomainName: [{}] Id:
[{}]", distribution.domainName(),
                distribution.id());
        }
        return distribution;
    }
}

```

- Para obtener más información sobre la API, consulte [CreateDistribution](#) la Referencia AWS SDK for Java 2.x de la API.

CreateFunction

En el siguiente ejemplo de código, se muestra cómo usar CreateFunction.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateFunction](#) la Referencia AWS SDK for Java 2.x de la API.

CreateKeyGroup

En el siguiente ejemplo de código, se muestra cómo usar CreateKeyGroup.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Un grupo de claves requiere al menos una clave pública que se utilice para verificar las URL o cookies firmadas.

```
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 obtener más información sobre la API, consulta [CreateKeyGroup](#) la Referencia AWS SDK for Java 2.x de la API.

CreatePublicKey

En el siguiente ejemplo de código, se muestra cómo usar CreatePublicKey.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

El siguiente ejemplo de código lee una clave pública y la carga en 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 obtener más información sobre la API, consulta [CreatePublicKey](#) la Referencia de AWS SDK for Java 2.x la API.

DeleteDistribution

En el siguiente ejemplo de código, se muestra cómo usar DeleteDistribution.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

El siguiente ejemplo de código actualiza una distribución a deshabilitada, utiliza un esperador que espera a que se implemente el cambio y, a continuación, elimina la distribución.

```

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 obtener más información sobre la API, consulta [DeleteDistribution](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateDistribution

En el siguiente ejemplo de código, se muestra cómo usar UpdateDistribution.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [UpdateDistribution](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Eliminar recursos de firma

En el siguiente ejemplo de código, se muestra cómo eliminar los recursos que se utilizan para acceder a contenido restringido de un bucket de Amazon Simple Storage Service (Amazon S3).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [DeleteKeyGroup](#)
 - [DeleteOriginAccessControl](#)
 - [DeletePublicKey](#)

Firmar URL y cookies

En el siguiente ejemplo de código, se muestra cómo crear URL firmadas y cookies que permitan el acceso a recursos restringidos.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la [CannedSignerRequest](#) clase para firmar las URL o las cookies con una política preestablecida.

```
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();
    }
}
```

Usa la [CustomSignerRequest](#) clase para firmar las URL o las cookies con una política personalizada. `activeDate` y `ipRange` son métodos opcionales.

```
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();
}
}

```

En el siguiente ejemplo, se muestra el uso de la [CloudFrontUtilities](#) clase para generar cookies y direcciones URL firmadas. [Vea](#) este ejemplo de código en [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 obtener más información sobre la API, consulte [CloudFrontUtilities](#) la referencia AWS SDK for Java 2.x de la API.

CloudWatch ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with CloudWatch.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

¿Hola CloudWatch

Los siguientes ejemplos de código muestran cómo empezar a usarlo CloudWatch.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListMetrics](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

DeleteAlarms

En el siguiente ejemplo de código, se muestra cómo usar DeleteAlarms.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteAlarms](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteAnomalyDetector

En el siguiente ejemplo de código, se muestra cómo usar DeleteAnomalyDetector.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
            .build();

        cw.deleteAnomalyDetector(request);
    }
}
```

```
        System.out.println("Successfully deleted the Anomaly Detector.");
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

- Para obtener más información sobre la API, consulta [DeleteAnomalyDetector](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteDashboards

En el siguiente ejemplo de código, se muestra cómo usar DeleteDashboards.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();
        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");

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

- Para obtener más información sobre la API, consulta [DeleteDashboards](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeAlarmHistory

En el siguiente ejemplo de código, se muestra cómo usar DescribeAlarmHistory.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
            .startDate(start)
            .endDate(endDate)
            .alarmName(alarmName)
            .historyItemType(HistoryItemType.ACTION)
            .build();

        DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
        List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
".");
        } else {
            for (AlarmHistoryItem item : historyItems) {
```

```
        System.out.println("History summary: " + item.historySummary());
        System.out.println("Time stamp: " + item.timestamp());
    }
}

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

- Para obtener más información sobre la API, consulta [DescribeAlarmHistory](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeAlarms

En el siguiente ejemplo de código, se muestra cómo usar DescribeAlarms.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
        }
    }
}
```

```
        System.out.println("Alarm description: " +
alarm.alarmDescription());
    }
} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Para obtener más información sobre la API, consulta [DescribeAlarms](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeAlarmsForMetric

En el siguiente ejemplo de código, se muestra cómo usar `DescribeAlarmsForMetric`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
```



```
        .namespace(customMetricNamespace)
        .build();

    while (!hasAlarm && retries > 0) {
        DescribeAlarmsForMetricResponse response =
    cw.describeAlarmsForMetric(metricRequest);
        hasAlarm = response.hasMetricAlarms();
        retries--;
        Thread.sleep(20000);
        System.out.println(".");
    }
    if (!hasAlarm)
        System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
    else
        System.out.println("Alarm state found for " + customMetricName +
".");

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

- Para obtener más información sobre la API, consulta [DescribeAlarmsForMetric](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeAnomalyDetectors

En el siguiente ejemplo de código, se muestra cómo usar DescribeAnomalyDetectors.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

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

- Para obtener más información sobre la API, consulta [DescribeAnomalyDetectors](#) la Referencia AWS SDK for Java 2.x de la API.

DisableAlarmActions

En el siguiente ejemplo de código, se muestra cómo usar `DisableAlarmActions`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DisableAlarmActions](#) la Referencia AWS SDK for Java 2.x de la API.

EnableAlarmActions

En el siguiente ejemplo de código, se muestra cómo usar `EnableAlarmActions`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [EnableAlarmActions](#) la Referencia AWS SDK for Java 2.x de la API.

GetMetricData

En el siguiente ejemplo de código, se muestra cómo usar GetMetricData.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
    }
}

```

```
Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
    ChronoUnit.MINUTES);

Metric met = Metric.builder()
    .metricName(customMetricName)
    .namespace(customMetricNamespace)
    .build();

MetricStat metStat = MetricStat.builder()
    .stat("Maximum")
    .period(1)
    .metric(met)
    .build();

MetricDataQuery dataQuery = MetricDataQuery.builder()
    .metricStat(metStat)
    .id("foo2")
    .returnData(true)
    .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

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

- Para obtener más información sobre la API, consulta [GetMetricData](#) la Referencia AWS SDK for Java 2.x de la API.

GetMetricStatistics

En el siguiente ejemplo de código, se muestra cómo usar `GetMetricStatistics`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
    String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
        .endTime(endDate)
        .startTime(start)
        .dimensions(myDimension)
        .metricName(metVal)
        .namespace(nameSpace)
        .period(86400)
        .statistics(Statistic.fromValue(metricOption))
        .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
```



```

                .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

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

```

- Para obtener más información sobre la API, consulta [GetMetricStatistics](#) la Referencia AWS SDK for Java 2.x de la API.

GetMetricWidgetImage

En el siguiente ejemplo de código, se muestra cómo usar `GetMetricWidgetImage`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {
        String myJSON = "{\n" +
            "  \"title\": \"Example Metric Graph\",\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 obtener más información sobre la API, consulta [GetMetricWidgetImage](#) la Referencia AWS SDK for Java 2.x de la API.

ListDashboards

En el siguiente ejemplo de código, se muestra cómo usar `ListDashboards`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });

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

- Para obtener más información sobre la API, consulta [ListDashboards](#) la Referencia AWS SDK for Java 2.x de la API.

ListMetrics

En el siguiente ejemplo de código, se muestra cómo usar `ListMetrics`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListMetrics](#) la Referencia AWS SDK for Java 2.x de la API.

PutAnomalyDetector

En el siguiente ejemplo de código, se muestra cómo usar PutAnomalyDetector.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
            .build();

        cw.putAnomalyDetector(anomalyDetectorRequest);
        System.out.println("Added anomaly detector for metric " +
customMetricName + ".");
    }
}
```

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

- Para obtener más información sobre la API, consulta [PutAnomalyDetector](#) la Referencia AWS SDK for Java 2.x de la API.

PutDashboard

En el siguiente ejemplo de código, se muestra cómo usar PutDashboard.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void createDashboardWithMetrics(CloudWatchClient cw, String  
dashboardName, String fileName) {  
    try {  
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()  
            .dashboardName(dashboardName)  
            .dashboardBody(readFileAsString(fileName))  
            .build();  
  
        PutDashboardResponse response = cw.putDashboard(dashboardRequest);  
        System.out.println(dashboardName + " was successfully created.");  
        List<DashboardValidationMessage> messages =  
response.dashboardValidationMessages();  
        if (messages.isEmpty()) {  
            System.out.println("There are no messages in the new Dashboard");  
        } else {  
            for (DashboardValidationMessage message : messages) {  
                System.out.println("Message is: " + message.message());  
            }  
        }  
    }  
}
```

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

- Para obtener más información sobre la API, consulta [PutDashboard](#) la Referencia AWS SDK for Java 2.x de la API.

PutMetricAlarm

En el siguiente ejemplo de código, se muestra cómo usar PutMetricAlarm.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createAlarm(CloudWatchClient cw, String fileName) {  
    try {  
        // Read values from the JSON file.  
        JsonParser parser = new JsonFactory().createParser(new File(fileName));  
        com.fasterxml.jackson.databind.JsonNode rootNode = new  
ObjectMapper().readTree(parser);  
        String customMetricNamespace =  
rootNode.findValue("customMetricNamespace").asText();  
        String customMetricName =  
rootNode.findValue("customMetricName").asText();  
        String alarmName = rootNode.findValue("exampleAlarmName").asText();  
        String emailTopic = rootNode.findValue("emailTopic").asText();  
        String accountId = rootNode.findValue("accountId").asText();  
        String region = rootNode.findValue("region").asText();  
  
        // Create a List for alarm actions.  
        List<String> alarmActions = new ArrayList<>();
```



```
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;

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

- Para obtener más información sobre la API, consulta [PutMetricAlarm](#) la Referencia AWS SDK for Java 2.x de la API.

PutMetricData

En el siguiente ejemplo de código, se muestra cómo usar PutMetricData.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1001.00)
            .timestamp(instant)
            .build();

        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
        metricDataList.add(datum2);
    }
}
```

```
PutMetricDataRequest request = PutMetricDataRequest.builder()
    .namespace(customMetricNamespace)
    .metricData(metricDataList)
    .build();

cw.putMetricData(request);
System.out.println("Added metric values for for metric " +
customMetricName);

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

- Para obtener más información sobre la API, consulta [PutMetricData](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Primeros pasos para usar las métricas, los paneles y las alarmas

En el siguiente ejemplo de código, se muestra cómo:

- Enumera los espacios de CloudWatch nombres y las métricas.
- Obtener estadísticas para una métrica y para la facturación estimada.
- Crear y actualizar un panel.
- Crear y agregar datos a una métrica.
- Crear y activar una alarma y, a continuación, consultar el historial de alarmas.
- Crear un detector de anomalías.
- Realice una imagen métrica y, luego, limpie los recursos.

SDK para Java 2.x

 Note

Hay más en marcha. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [DeleteAlarms](#)
 - [DeleteAnomalyDetector](#)
 - [DeleteDashboards](#)
 - [DescribeAlarmHistory](#)
 - [DescribeAlarms](#)
 - [DescribeAlarmsForMetric](#)
 - [DescribeAnomalyDetectors](#)
 - [GetMetricData](#)

- [GetMetricStatistics](#)
- [GetMetricWidgetImage](#)
- [ListMetrics](#)
- [PutAnomalyDetector](#)
- [PutDashboard](#)
- [PutMetricAlarm](#)
- [PutMetricData](#)

CloudWatch Ejemplos de eventos con SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Java 2.x with CloudWatch Events.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

PutEvents

En el siguiente ejemplo de código, se muestra cómo usar PutEvents.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [PutEvents](#) la Referencia AWS SDK for Java 2.x de la API.

PutRule

En el siguiente ejemplo de código, se muestra cómo usar PutRule.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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",
            ruleArn, response.ruleArn());

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

- Para obtener más información sobre la API, consulta [PutRule](#) la Referencia AWS SDK for Java 2.x de la API.

PutTargets

En el siguiente ejemplo de código, se muestra cómo usar PutTargets.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [PutTargets](#) la Referencia AWS SDK for Java 2.x de la API.

CloudWatch Ejemplos de registros mediante SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso AWS SDK for Java 2.x de CloudWatch registros.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

DeleteSubscriptionFilter

En el siguiente ejemplo de código, se muestra cómo usar DeleteSubscriptionFilter.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteSubscriptionFilter](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeSubscriptionFilters

En el siguiente ejemplo de código, se muestra cómo usar DescribeSubscriptionFilters.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeSubscriptionFilters](#) la Referencia AWS SDK for Java 2.x de la API.

PutSubscriptionFilter

En el siguiente ejemplo de código, se muestra cómo usar PutSubscriptionFilter.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [PutSubscriptionFilter](#) la Referencia AWS SDK for Java 2.x de la API.

StartLiveTail

En el siguiente ejemplo de código, se muestra cómo usar StartLiveTail.

SDK para Java 2.x

Incluir los archivos requeridos.

```
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;
```

Gestione los eventos de la sesión de 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 la sesión de 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));

```

Detenga la sesión de Live Tail una vez transcurrido un periodo de tiempo.

```

/* 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 obtener más información sobre la API, consulte [StartLiveTail](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de identidad de Amazon Cognito usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante Amazon Cognito Identity. AWS SDK for Java 2.x

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

CreateIdentityPool

En el siguiente ejemplo de código, se muestra cómo usar CreateIdentityPool.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta [CreateIdentityPool](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteIdentityPool

En el siguiente ejemplo de código, se muestra cómo usar DeleteIdentityPool.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteIdentityPool](#) la Referencia AWS SDK for Java 2.x de la API.

GetCredentialsForIdentity

En el siguiente ejemplo de código, se muestra cómo usar `GetCredentialsForIdentity`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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 obtener más información sobre la API, consulta [GetCredentialsForIdentity](#) la Referencia AWS SDK for Java 2.x de la API.

ListIdentityPools

En el siguiente ejemplo de código, se muestra cómo usar `ListIdentityPools`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.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 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 obtener más información sobre la API, consulta [ListIdentityPools](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Amazon Cognito Identity Provider usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante el AWS SDK for Java 2.x uso de Amazon Cognito Identity Provider.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Amazon Cognito

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon Cognito.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListUserPools](#) la Referencia AWS SDK for Java 2.x de la API.

Temas


- [Acciones](#)
- [Escenarios](#)

Acciones

AdminGetUser

En el siguiente ejemplo de código, se muestra cómo usar AdminGetUser.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [AdminGetUser](#) la Referencia AWS SDK for Java 2.x de la API.

AdminInitiateAuth

En el siguiente ejemplo de código, se muestra cómo usar AdminInitiateAuth.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
             String clientId, String userName, String password, String userPoolId) {
    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                        .clientId(clientId)
                        .userPoolId(userPoolId)
                        .authParameters(authParameters)
                        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                        .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName());
        return response;

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

    return null;
}
```

- Para obtener más información sobre la API, consulta [AdminInitiateAuth](#) la Referencia AWS SDK for Java 2.x de la API.

AdminRespondToAuthChallenge

En el siguiente ejemplo de código, se muestra cómo usar `AdminRespondToAuthChallenge`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
    String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
        .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
        + respondToAuthChallengeResult.authenticationResult());
}
```

- Para obtener más información sobre la API, consulta [AdminRespondToAuthChallenge](#) la Referencia AWS SDK for Java 2.x de la API.

AssociateSoftwareToken

En el siguiente ejemplo de código, se muestra cómo usar AssociateSoftwareToken.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
        .session(session)
        .build();

    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
        .associateSoftwareToken(softwareTokenRequest);
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

- Para obtener más información sobre la API, consulta [AssociateSoftwareToken](#) la Referencia AWS SDK for Java 2.x de la API.

ConfirmSignUp

En el siguiente ejemplo de código, se muestra cómo usar ConfirmSignUp.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
    String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

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

- Para obtener más información sobre la API, consulta [ConfirmSignUp](#) la Referencia AWS SDK for Java 2.x de la API.

CreateUserPool

En el siguiente ejemplo de código, se muestra cómo usar CreateUserPool.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateUserPool](#) la Referencia AWS SDK for Java 2.x de la API.

CreateUserPoolClient

En el siguiente ejemplo de código, se muestra cómo usar CreateUserPoolClient.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateUserPoolClient](#) la Referencia AWS SDK for Java 2.x de la API.

ListUserPools

En el siguiente ejemplo de código, se muestra cómo usar `ListUserPools`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListUserPools](#) la Referencia AWS SDK for Java 2.x de la API.

ListUsers

En el siguiente ejemplo de código, se muestra cómo usar ListUsers.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListUsers](#) la Referencia AWS SDK for Java 2.x de la API.

ResendConfirmationCode

En el siguiente ejemplo de código, se muestra cómo usar `ResendConfirmationCode`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

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

- Para obtener más información sobre la API, consulta [ResendConfirmationCode](#) la Referencia AWS SDK for Java 2.x de la API.

SignUp

En el siguiente ejemplo de código, se muestra cómo usar SignUp.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);
    try {
        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
            .username(userName)
            .clientId(clientId)
            .password(password)
            .build();

        identityProviderClient.signUp(signUpRequest);
        System.out.println("User has been signed up ");
    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [SignUp](#) la Referencia AWS SDK for Java 2.x de la API.

VerifySoftwareToken

En el siguiente ejemplo de código, se muestra cómo usar `VerifySoftwareToken`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

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

- Para obtener más información sobre la API, consulta [VerifySoftwareToken](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Registro de un usuario en un grupo de usuarios que requiera MFA

En el siguiente ejemplo de código, se muestra cómo:

- Registre y confirme a un usuario con un nombre de usuario, una contraseña y una dirección de correo electrónico.
- Configure la autenticación multifactor asociando una aplicación MFA al usuario.
- Inicie sesión con una contraseña y un código MFA.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.CognitoIdentityProviderException;
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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [AdminGetUser](#)
 - [AdminInitiateAuth](#)
 - [AdminRespondToAuthChallenge](#)
 - [AssociateSoftwareToken](#)
 - [ConfirmDevice](#)
 - [ConfirmSignUp](#)
 - [InitiateAuth](#)
 - [ListUsers](#)
 - [ResendConfirmationCode](#)
 - [RespondToAuthChallenge](#)
 - [SignUp](#)
 - [VerifySoftwareToken](#)

Ejemplos de Amazon Comprehend usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x con Amazon Comprehend.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

CreateDocumentClassifier

En el siguiente ejemplo de código, se muestra cómo usar `CreateDocumentClassifier`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateDocumentClassifier](#) la Referencia AWS SDK for Java 2.x de la API.

DetectDominantLanguage

En el siguiente ejemplo de código, se muestra cómo usar DetectDominantLanguage.

SDK para Java 2.x

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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 obtener más información sobre la API, consulta [DetectDominantLanguage](#) la Referencia AWS SDK for Java 2.x de la API.

DetectEntities

En el siguiente ejemplo de código, se muestra cómo usar DetectEntities.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DetectEntities](#) la Referencia AWS SDK for Java 2.x de la API.

DetectKeyPhrases

En el siguiente ejemplo de código, se muestra cómo usar DetectKeyPhrases.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DetectKeyPhrases](#) la Referencia AWS SDK for Java 2.x de la API.

DetectSentiment

En el siguiente ejemplo de código, se muestra cómo usar DetectSentiment.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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 obtener más información sobre la API, consulta [DetectSentiment](#) la Referencia AWS SDK for Java 2.x de la API.

DetectSyntax

En el siguiente ejemplo de código, se muestra cómo usar DetectSyntax.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DetectSyntax](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de DynamoDB usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante DynamoDB. AWS SDK for Java 2.x

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a DynamoDB

En los siguientes ejemplos de código, se muestra cómo empezar a utilizar DynamoDB.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListTables](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)
- [Ejemplos sin servidor](#)

Acciones

BatchGetItem

En el siguiente ejemplo de código, se muestra cómo usar BatchGetItem.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

muestra cómo obtener elementos por lotes mediante el cliente de servicio.

```
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.");
        }
    }
}

```

muestra cómo obtener elementos por lotes mediante el cliente de servicio y un 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 obtener más información sobre la API, consulta [BatchGetItem](#) la Referencia AWS SDK for Java 2.x de la API.

BatchWriteItem

En el siguiente ejemplo de código, se muestra cómo usar BatchWriteItem.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Inserta muchos elementos en una tabla con el cliente de servicio.

```
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(item1Attributes)

        // 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(item2Attributes)

            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);
    }
}
}
```

Inserta muchos elementos en una tabla mediante el cliente mejorado.

```
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 obtener más información sobre la API, consulta [BatchWriteItem](#) la Referencia AWS SDK for Java 2.x de la API.

CreateTable

En el siguiente ejemplo de código, se muestra cómo usar CreateTable.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateTable](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteItem

En el siguiente ejemplo de código, se muestra cómo usar DeleteItem.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteItem](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteTable

En el siguiente ejemplo de código, se muestra cómo usar DeleteTable.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteTable](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeTable

En el siguiente ejemplo de código, se muestra cómo usar DescribeTable.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeTable](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeTimeToLive

En el siguiente ejemplo de código, se muestra cómo usar `DescribeTimeToLive`.

SDK para Java 2.x

Se describe la configuración de TTL en una tabla de DynamoDB existente.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.Optional;

    final DescribeTimeToLiveRequest request =
DescribeTimeToLiveRequest.builder()
        .tableName(tableName)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final DescribeTimeToLiveResponse response =
ddb.describeTimeToLive(request);
        System.out.println(tableName + " description of time to live is "
            + response.toString());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    }

```

```
    } catch (DynamoDbException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
    System.exit(0);
```

- Para obtener más información sobre la API, consulte [DescribeTimeToLive](#) la Referencia AWS SDK for Java 2.x de la API.

GetItem

En el siguiente ejemplo de código, se muestra cómo usar `GetItem`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtiene un elemento de una mesa mediante el `DynamoDbClient`.

```
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 obtener más información sobre la API, consulte [GetItem](#) la Referencia AWS SDK for Java 2.x de la API.

ListTables

En el siguiente ejemplo de código, se muestra cómo usar `ListTables`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListTables](#) la Referencia AWS SDK for Java 2.x de la API.

PutItem

En el siguiente ejemplo de código, se muestra cómo usar `PutItem`.

SDK para Java 2.x

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Coloca un objeto en una mesa 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 obtener más información sobre la API, consulte [PutItem](#) la Referencia AWS SDK for Java 2.x de la API.

Query

En el siguiente ejemplo de código, se muestra cómo usar Query.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Consulta una tabla mediante [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.ExpressionAttributeName.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;
}
}

```

Consultar una tabla con `DynamoDbClient` y un índice secundario.

```

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 obtener información sobre la API, consulte [Query](#) en la referencia de la API de AWS SDK for Java 2.x .

Scan

En el siguiente ejemplo de código, se muestra cómo usar Scan.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Escanea una tabla de Amazon DynamoDB utilizando. [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.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 obtener información sobre la API, consulte [Scan](#) en la referencia de la API de AWS SDK for Java 2.x .

UpdateItem

En el siguiente ejemplo de código, se muestra cómo usar UpdateItem.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Actualiza un elemento de una tabla mediante [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 obtener más información sobre la API, consulte [UpdateItem](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateTimeToLive

En el siguiente ejemplo de código, se muestra cómo usar `UpdateTimeToLive`.

SDK para Java 2.x

Habilite TTL en una tabla de DynamoDB existente.

```
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.ResourceNotFoundException;
```



```

import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.Optional;

    final TimeToLiveSpecification ttlSpecification =
TimeToLiveSpecification.builder()
    .attributeName(ttlAttributeName)
    .enabled(true)
    .build();
    final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
    .tableName(tableName)
    .timeToLiveSpecification(ttlSpecification)
    .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
        final UpdateTimeToLiveResponse response =
ddb.updateTimeToLive(request);
        System.out.println(tableName + " had its TTL 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.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done!");

```

Deshabilite TTL en una tabla de DynamoDB existente.

```

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.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

```

```

import java.util.Optional;

    final Region region = Optional.ofNullable(args[2]).isEmpty() ?
Region.US_EAST_1 : Region.of(args[2]);
    final TimeToLiveSpecification ttlSpecification =
TimeToLiveSpecification.builder()
        .attributeName(ttlAttributeName)
        .enabled(false)
        .build();
    final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
        .tableName(tableName)
        .timeToLiveSpecification(ttlSpecification)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final UpdateTimeToLiveResponse response = ddb.updateTimeToLive(request);
        System.out.println(tableName + " had its TTL 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.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done!");

```

- Para obtener más información sobre la API, consulte [UpdateTimeToLive](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Actualización condicional del TTL de un elemento

El siguiente ejemplo de código muestra cómo actualizar condicionalmente el TTL de un elemento.

SDK para Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.amazon.samplelib.ttl;

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.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

public class UpdateTTLConditional {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <tableName> <primaryKey> <sortKey> <newTtlAttribute> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
or partition key.
                sortKey - The name of the sort key. Also known as the range
attribute.
                newTtlAttribute - New attribute name (as part of the update
command)
                region (optional) - The AWS region that the Amazon DynamoDB
table is located in. (Default: us-east-1)
            """;
        // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
        if (!(args.length == 4 || args.length == 5)) {
            System.out.println(usage);
            System.exit(1);
        }
        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final String newTtlAttribute = args[3];
```

```

    Region region = Optional.ofNullable(args[4]).isEmpty() ? Region.US_EAST_1 :
    Region.of(args[4]);

    // Get current time in epoch second format
    final long currentTime = System.currentTimeMillis() / 1000;
    // Calculate expiration time 90 days from now in epoch second format
    final long expireDate = currentTime + (90 * 24 * 60 * 60);
    // An expression that defines one or more attributes to be updated, the
    action to be performed on them, and new values for them.
    final String updateExpression = "SET newTtlAttribute = :val1";
    // A condition that must be satisfied in order for a conditional update to
    succeed.
    final String conditionExpression = "expireAt > :val2";

    final ImmutableMap<String, AttributeValue> keyMap =
        ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
            "sortKey", AttributeValue.fromS(sortKey));
    final Map<String, AttributeValue> expressionAttributeValues =
    ImmutableMap.of(
        ":val1", AttributeValue.builder().s(newTtlAttribute).build(),
        ":val2",
    AttributeValue.builder().s(String.valueOf(expireDate)).build()
    );

    final UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(keyMap)
        .updateExpression(updateExpression)
        .conditionExpression(conditionExpression)
        .expressionAttributeValues(expressionAttributeValues)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final UpdateItemResponse response = ddb.updateItem(request);
        System.out.println(tableName + " UpdateItem operation with conditional
    TTL successful. 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.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
    }

```

```
        System.exit(1);
    }
    System.exit(0);
}
}
```

- Para obtener más información sobre la API, consulta la Referencia de [UpdateItem](#) la AWS SDK for Java 2.x API.

Creación de un elemento con un TTL

El siguiente ejemplo de código muestra cómo crear un elemento con TTL.

SDK para Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

package com.amazon.samplelib.ttl;

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 software.amazon.awssdk.utils.ImmutableMap;

import java.io.Serializable;
import java.util.Map;
import java.util.Optional;

public class CreateTTL {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
                or partition key.
```

```

        sortKey - The name of the sort key. Also known as the range
attribute.
        region (optional) - The AWS region that the Amazon DynamoDB
table is located in. (Default: us-east-1)
        """;
    // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
    if (!(args.length == 3 || args.length == 4)) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String primaryKey = args[1];
    String sortKey = args[2];
    Region region = Optional.ofNullable(args[3]).isEmpty() ? Region.US_EAST_1 :
Region.of(args[3]);

    // Get current time in epoch second format
    final long createDate = System.currentTimeMillis() / 1000;

    // Calculate expiration time 90 days from now in epoch second format
    final long expireDate = createDate + (90 * 24 * 60 * 60);

    final ImmutableMap<String, ? extends Serializable> itemMap =
        ImmutableMap.of("primaryKey", primaryKey,
            "sortKey", sortKey,
            "creationDate", createDate,
            "expireAt", expireDate);
    final PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item((Map<String, AttributeValue>) itemMap)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final PutItemResponse response = ddb.putItem(request);
        System.out.println(tableName + " PutItem operation with TTL successful.
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.exit(1);
    }

```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.exit(0);
    }
}
```

- Para obtener más información sobre la API, consulte [PutItem](#) la Referencia de AWS SDK for Java 2.x la API.

Introducción a tablas, elementos y consultas

En el siguiente ejemplo de código, se muestra cómo:

- Creación de una tabla que pueda contener datos de películas.
- Colocar, obtener y actualizar una sola película en la tabla.
- Escribir los datos de películas en la tabla a partir de un archivo JSON de ejemplo.
- Consultar películas que se hayan estrenado en un año determinado.
- Buscar películas que se hayan estrenado en un intervalo de años.
- Eliminación de una película de la tabla y, a continuación, eliminar la tabla.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear una tabla de 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);
    }
}

```

Crear una función auxiliar para descargar y extraer el archivo JSON de muestra.

```

// 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++;
    }
}
```

Obtener un elemento de una tabla.

```
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);
    }
}
```

```
    }  
}
```

Ejemplo 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)
 - [PutItem](#)
 - [Query](#)
 - [Scan](#)
 - [UpdateItem](#)

Consultar una tabla mediante lotes de instrucciones PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtención de un lote de elementos mediante la ejecución de varias instrucciones SELECT.
- Agregar un lote de elementos mediante la ejecución de varias instrucciones INSERT.
- Actualizar un lote de elementos con la ejecución de varias instrucciones UPDATE.
- Eliminación de un lote de elementos con la ejecución de varias instrucciones DELETE.

SDK para Java 2.x

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQBatch";
        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 obtener más información sobre la API, consulta [BatchExecuteStatement](#) la Referencia AWS SDK for Java 2.x de la API.

Consultar una tabla con PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtención de un artículo mediante una instrucción SELECT.
- Agregar un elemento mediante una instrucción INSERT.
- Actualizar un elemento mediante una instrucción UPDATE.
- Eliminación de un elemento mediante una instrucción DELETE.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ExecuteStatement](#) la Referencia AWS SDK for Java 2.x de la API.

Consulta de elementos de TTL

El siguiente ejemplo de código muestra cómo consultar elementos TTL.

SDK para Java 2.x

Consulte una expresión filtrada para recopilar elementos de TTL en una tabla de DynamoDB.

```
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 software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

// Get current time in epoch second format (comparing against expiry
attribute)
final long currentTime = System.currentTimeMillis() / 1000;

// A string that contains conditions that DynamoDB applies after the Query
operation, but before the data is returned to you.
final String keyConditionExpression = "#pk = :pk";

// The condition that specifies the key values for items to be retrieved by
the Query action.
final String filterExpression = "#ea > :ea";
final Map<String, String> expressionAttributeNames = ImmutableMap.of(
    "#pk", "primaryKey",
    "#ea", "expireAt");
final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
    ":pk", AttributeValue.builder().s(primaryKey).build(),
    ":ea",
AttributeValue.builder().s(String.valueOf(currentTime)).build()
);

final QueryRequest request = QueryRequest.builder()
    .tableName(tableName)
    .keyConditionExpression(keyConditionExpression)
    .filterExpression(filterExpression)
    .expressionAttributeNames(expressionAttributeNames)
    .expressionAttributeValues(expressionAttributeValues)
    .build();
```

```
try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
    final QueryResponse response = ddb.query(request);
    System.out.println(tableName + " Query operation with TTL successful.
Request id is "
        + response.responseMetadata().requestId());
    // Print the items that are not expired
    for (Map<String, AttributeValue> item : response.items()) {
        System.out.println(item.toString());
    }
} catch (ResourceNotFoundException e) {
    System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
    System.exit(1);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.exit(0);
```

- Para obtener información sobre la API, consulte [Query](#) en la referencia de la API de AWS SDK for Java 2.x .

Actualización del TTL de un elemento

El siguiente ejemplo de código muestra cómo actualizar el TTL de un elemento.

SDK para Java 2.x

Actualice el TTL en un elemento de DynamoDB existente en una tabla.

```
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.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
```

```
import java.util.Optional;

// Get current time in epoch second format
final long currentTime = System.currentTimeMillis() / 1000;
// Calculate expiration time 90 days from now in epoch second format
final long expireDate = currentTime + (90 * 24 * 60 * 60);
// An expression that defines one or more attributes to be updated, the
action to be performed on them, and new values for them.
final String updateExpression = "SET updatedAt=:c, expireAt=:e";

final ImmutableMap<String, AttributeValue> keyMap =
    ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
        "sortKey", AttributeValue.fromS(sortKey));
final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
    ":c",
AttributeValue.builder().s(String.valueOf(currentTime)).build(),
    ":e", AttributeValue.builder().s(String.valueOf(expireDate)).build()
);

final UpdateItemRequest request = UpdateItemRequest.builder()
    .tableName(tableName)
    .key(keyMap)
    .updateExpression(updateExpression)
    .expressionAttributeValues(expressionAttributeValues)
    .build();
try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
    final UpdateItemResponse response = ddb.updateItem(request);
    System.out.println(tableName + " UpdateItem operation with TTL
successful. 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.exit(1);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.exit(0);
```

- Para obtener más información sobre la API, consulta [UpdateItem](#) la Referencia de AWS SDK for Java 2.x la API.

Ejemplos sin servidor

Invocación de una función de Lambda desde un desencadenador de DynamoDB

El siguiente ejemplo de código muestra cómo implementar una función de Lambda que recibe un evento desencadenado al recibir registros de una transmisión de DynamoDB. La función recupera la carga útil de DynamoDB y registra el contenido del registro.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Consumo de un evento de DynamoDB con Lambda mediante Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import
    com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
```



```
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " + GSON.toJson(record.getDynamodb()));
    }
}
```

Notificación de los errores de los elementos del lote de las funciones de Lambda con un desencadenador de DynamoDB

El siguiente ejemplo de código muestra cómo implementar una respuesta por lotes parcial para las funciones de Lambda que reciben eventos de una transmisión de DynamoDB. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Java 2.x

Note

Hay más información al respecto. [GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.](#)

Notificación de los errores de los elementos del lote de DynamoDB con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
    Serializable> {

    @Override
```

```
public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
{
    List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
    ArrayList<>();
    String curRecordSequenceNumber = "";

    for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
    input.getRecords()) {
        try {
            //Process your record
            StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
            curRecordSequenceNumber = dynamodbRecord.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();
}
```

Ejemplos de Amazon EC2 usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Amazon EC2.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Amazon EC2

En los siguientes ejemplos de código, se muestra cómo empezar a utilizar Amazon EC2.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeSecurityGroups](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

AllocateAddress

En el siguiente ejemplo de código, se muestra cómo usar AllocateAddress.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [AllocateAddress](#) la Referencia AWS SDK for Java 2.x de la API.

AssociateAddress

En el siguiente ejemplo de código, se muestra cómo usar AssociateAddress.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Para obtener más información sobre la API, consulta [AssociateAddress](#) la Referencia AWS SDK for Java 2.x de la API.

AuthorizeSecurityGroupIngress

En el siguiente ejemplo de código, se muestra cómo usar AuthorizeSecurityGroupIngress.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
```

```

        .ipPermissions(ipPerm, ipPerm2)
        .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

```

- Para obtener más información sobre la API, consulta [AuthorizeSecurityGroupIngress](#) la Referencia AWS SDK for Java 2.x de la API.

CreateKeyPair

En el siguiente ejemplo de código, se muestra cómo usar CreateKeyPair.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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 obtener más información sobre la API, consulta [CreateKeyPair](#) la Referencia AWS SDK for Java 2.x de la API.

CreateSecurityGroup

En el siguiente ejemplo de código, se muestra cómo usar CreateSecurityGroup.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
```



```
        .build();

    IpPermission ipPerm = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(80)
        .fromPort(80)
        .ipRanges(ipRange)
        .build();

    IpPermission ipPerm2 = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(22)
        .fromPort(22)
        .ipRanges(ipRange)
        .build();

    AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(groupName)
        .ipPermissions(ipPerm, ipPerm2)
        .build();

    ec2.authorizeSecurityGroupIngress(authRequest);
    System.out.println("Successfully added ingress policy to security group
" + groupName);
    return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Para obtener más información sobre la API, consulta [CreateSecurityGroup](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteKeyPair

En el siguiente ejemplo de código, se muestra cómo usar DeleteKeyPair.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteKeyPair](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteSecurityGroup

En el siguiente ejemplo de código, se muestra cómo usar DeleteSecurityGroup.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteSecurityGroup](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeInstanceTypes

En el siguiente ejemplo de código, se muestra cómo usar DescribeInstanceTypes.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// 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 obtener más información sobre la API, consulta [DescribeInstanceTypes](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeInstances

En el siguiente ejemplo de código, se muestra cómo usar DescribeInstances.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeInstances](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeKeyPairs

En el siguiente ejemplo de código, se muestra cómo usar DescribeKeyPairs.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeKeyPairs](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeSecurityGroups

En el siguiente ejemplo de código, se muestra cómo usar DescribeSecurityGroups.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeSecurityGroups](#) la Referencia AWS SDK for Java 2.x de la API.

DisassociateAddress

En el siguiente ejemplo de código, se muestra cómo usar `DisassociateAddress`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [DisassociateAddress](#) la Referencia AWS SDK for Java 2.x de la API.

ReleaseAddress

En el siguiente ejemplo de código, se muestra cómo usar `ReleaseAddress`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ReleaseAddress](#) la Referencia AWS SDK for Java 2.x de la API.

RunInstances

En el siguiente ejemplo de código, se muestra cómo usar RunInstances.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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.waiter().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 obtener más información sobre la API, consulta [RunInstances](#) la Referencia AWS SDK for Java 2.x de la API.

StartInstances

En el siguiente ejemplo de código, se muestra cómo usar StartInstances.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void startInstance(Ec2Client ec2, String instanceId) {  
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()  
        .overrideConfiguration(b -> b.maxAttempts(100))  
        .client(ec2)  
        .build();  
  
    StartInstancesRequest request = StartInstancesRequest.builder()  
        .instanceIds(instanceId)  
        .build();  
  
    System.out.println("Use an Ec2Waiter to wait for the instance to run. This  
will take a few minutes.");  
    ec2.startInstances(request);  
    DescribeInstancesRequest instanceRequest =  
DescribeInstancesRequest.builder()  
        .instanceIds(instanceId)  
        .build();  
  
    WaiterResponse<DescribeInstancesResponse> waiterResponse =  
ec2Waiter.waitUntilInstanceRunning(instanceRequest);  
    waiterResponse.matched().response().ifPresent(System.out::println);  
    System.out.println("Successfully started instance " + instanceId);  
}
```

```
}
```

- Para obtener más información sobre la API, consulta [StartInstances](#) la Referencia AWS SDK for Java 2.x de la API.

StopInstances

En el siguiente ejemplo de código, se muestra cómo usar StopInstances.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}
```

- Para obtener más información sobre la API, consulta [StopInstances](#) la Referencia AWS SDK for Java 2.x de la API.

TerminateInstances

En el siguiente ejemplo de código, se muestra cómo usar TerminateInstances.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [TerminateInstances](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Creación y administración de un servicio resiliente

El siguiente ejemplo de código muestra cómo crear un servicio web con equilibrio de carga que muestre recomendaciones de libros, películas y canciones. El ejemplo muestra cómo responde el servicio a los errores y cómo reestructurarlo para aumentar la resiliencia cuando se produzcan errores.

- Utilice un grupo de Amazon EC2 Auto Scaling para crear instancias de Amazon Elastic Compute Cloud (Amazon EC2) basadas en una plantilla de lanzamiento y para mantener el número de instancias dentro de un rango específico.
- Administre y distribuya las solicitudes HTTP con Elastic Load Balancing.
- Supervise el estado de las instancias de un grupo de escalado automático y reenvíe las solicitudes solo a las instancias en buen estado.
- Ejecute un servidor web Python en cada instancia de EC2 para administrar las solicitudes HTTP. El servidor web responde con recomendaciones y comprobaciones de estado.
- Simule un servicio de recomendaciones con una tabla de Amazon DynamoDB.
- Controle la respuesta del servidor web a las solicitudes y las comprobaciones de estado actualizando AWS Systems Manager los parámetros.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute el escenario interactivo en un símbolo del sistema.

```
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("""
check to update
                                Note that it can take a minute or two for the health
                                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);
}
}
```

Cree una clase que agrupe las acciones de escalado automático y Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
```

```
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
```

```
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                // name.
    .build();

// Replace the IAM instance profile association for the EC2 instance.
ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}

System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    }
}
```

```

    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.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")))
        Collections.singletonList("cd / && sudo python3 server.py

    .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.");
}
}
```


Cree una clase que resuma las acciones de 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 "";
}
}
```

Cree una clase que utilice DynamoDB para simular un servicio de recomendaciones.

```
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
     * 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);
    }
}
```

Cree una clase que agrupe las acciones de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de 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](#)

Comience a utilizar instancias

En el siguiente ejemplo de código, se muestra cómo:

- Cree un par de claves y un grupo de seguridad.
- Seleccione una Imagen de máquina de Amazon (AMI) y un tipo de instancia; a continuación, cree una instancia.
- Detenga y vuelva a iniciar la instancia.

- Asocie una dirección IP elástica a su instancia.
- Conéctese a tu instancia con SSH y, a continuación, limpie los recursos.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.waitForInstanceRunning(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.waitForInstanceStopped(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.getInstanceTypes();
        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.getInstanceType().toString());
            instanceType = type.getInstanceType().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
(ssoftware.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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de 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](#)

Ejemplos de Amazon ECR con el SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Amazon ECR.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Hola, Amazon ECR

Los siguientes ejemplos de código muestran cómo empezar a utilizar Amazon ECR.

SDK para Java 2.x

 Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;

public class HelloECR {

    public static void main(String[] args) {
        final String usage = ""
            Usage:    <repositoryName>

            Where:
                repositoryName - The name of the Amazon ECR repository.
            """;

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

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }

    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();
```

```
ListImagesIterable imagesIterable =
ecrClient.listImagesPaginator(listImagesPaginator);
imagesIterable.stream()
    .flatMap(r -> r.imageIds().stream())
    .forEach(image -> System.out.println("The docker image tag is: "
+image.imageTag()));
}
```

- Para obtener más información sobre la API, consulta [ListImages en AWS SDK for Java 2.x la referencia](#) de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateRepository

En el siguiente ejemplo de código, se muestra cómo usar CreateRepository.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/**
 * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
 *
 * @param repoName the name of the repository to create.
 * @return the Amazon Resource Name (ARN) of the created repository, or an empty
 string if the operation failed.
 * @throws IllegalArgumentException If repository name is invalid.
 * @throws RuntimeException if an error occurs while creating the
 repository.
```

```
    */
    public String createECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        CreateRepositoryRequest request = CreateRepositoryRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
successfully.");
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");

                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();

                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }
}
```

- Para obtener más información sobre la API, consulta [CreateRepository](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteRepository

En el siguiente ejemplo de código, se muestra cómo usar DeleteRepository.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/**
 * Deletes an ECR (Elastic Container Registry) repository.
 *
 * @param repoName the name of the repository to delete.
 * @throws IllegalArgumentException if the repository name is null or empty.
 * @throws EcrException if there is an error deleting the repository.
 * @throws RuntimeException if an unexpected error occurs during the deletion
 process.
 */
public void deleteECRRepository(String repoName) {
    if (repoName == null || repoName.isEmpty()) {
        throw new IllegalArgumentException("Repository name cannot be null or
empty");
    }

    DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
        .force(true)
        .repositoryName(repoName)
        .build();

    CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
    response.whenComplete((deleteRepositoryResponse, ex) -> {
```

```
        if (deleteRepositoryResponse != null) {
            System.out.println("You have successfully deleted the " + repoName +
" repository");
        } else {
            Throwable cause = ex.getCause();
            if (cause instanceof EcrException) {
                throw (EcrException) cause;
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        }
    });

    // Wait for the CompletableFuture to complete
    response.join();
}
```

- Para obtener más información sobre la API, consulta [DeleteRepository](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeImages

En el siguiente ejemplo de código, se muestra cómo usar DescribeImages.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag       The tag of the image to verify.
```

```

    * @throws EcrException          if there is an error retrieving the image
information from Amazon ECR.
    * @throws CompletionException  if the asynchronous operation completes
exceptionally.
    */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }

```

- Para obtener más información sobre la API, consulta [Describelmages](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeRepositories

En el siguiente ejemplo de código, se muestra cómo usar DescribeRepositories.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/**
 * Retrieves the repository URI for the specified repository name.
 *
 * @param repoName the name of the repository to retrieve the URI for.
 * @return the repository URI for the specified repository name.
 * @throws EcrException if there is an error retrieving the repository
information.
 * @throws CompletionException if the asynchronous operation completes
exceptionally.
 */
public void getRepositoryURI(String repoName) {
    DescribeRepositoriesRequest request = DescribeRepositoriesRequest.builder()
        .repositoryNames(repoName)
        .build();

    CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
    response.whenComplete((describeRepositoriesResponse, ex) -> {
        if (ex != null) {
            Throwable cause = ex.getCause();
            if (cause instanceof InterruptedException) {
                Thread.currentThread().interrupt();
                String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                throw new RuntimeException(errorMessage, cause);
            } else if (cause instanceof EcrException) {
                throw (EcrException) cause;
            } else {
                String errorMessage = "Unexpected error: " + cause.getMessage();
                throw new RuntimeException(errorMessage, cause);
            }
        }
    })
}
```

```

        } else {
            if (describeRepositoriesResponse != null) {
                if (!describeRepositoriesResponse.repositories().isEmpty()) {
                    String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                    System.out.println("Repository URI found: " +
repositoryUri);
                } else {
                    System.out.println("No repositories found for the given
name.");
                }
            } else {
                System.err.println("No response received from
describeRepositories.");
            }
        }
    });
    response.join();
}

```

- Para obtener más información sobre la API, consulta [DescribeRepositories](#) la Referencia AWS SDK for Java 2.x de la API.

GetAuthorizationToken

En el siguiente ejemplo de código, se muestra cómo usar GetAuthorizationToken.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/**
 * Retrieves the authorization token for Amazon Elastic Container Registry
 (ECR).
 * This method makes an asynchronous call to the ECR client to retrieve the
 authorization token.

```

```

    * If the operation is successful, the method prints the token to the console.
    * If an exception occurs, the method handles the exception and prints the error
message.
    *
    * @throws EcrException      if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
    public void getAuthToken() {
        CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
        response.whenComplete((authorizationTokenResponse, ex) -> {
            if (authorizationTokenResponse != null) {
                AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
                String token = authorizationData.authorizationToken();
                if (!token.isEmpty()) {
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }
}

```

- Para obtener más información sobre la API, consulta [GetAuthorizationToken](#) la Referencia AWS SDK for Java 2.x de la API.

GetRepositoryPolicy

En el siguiente ejemplo de código, se muestra cómo usar `GetRepositoryPolicy`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/**
 * Gets the repository policy for the specified repository.
 *
 * @param repoName the name of the repository.
 * @throws EcrException if an AWS error occurs while getting the repository
 policy.
 */
public String getRepoPolicy(String repoName) {
    if (repoName == null || repoName.isEmpty()) {
        throw new IllegalArgumentException("Repository name cannot be null or
empty");
    }

    GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
        .repositoryName(repoName)
        .build();

    CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
    response.whenComplete((resp, ex) -> {
        if (resp != null) {
            System.out.println("Repository policy retrieved successfully.");
        } else {
            if (ex.getCause() instanceof EcrException) {
                throw (EcrException) ex.getCause();
            } else {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        }
    });
}
```

```

    GetRepositoryPolicyResponse result = response.join();
    return result != null ? result.policyText() : null;
}

```

- Para obtener más información sobre la API, consulta [GetRepositoryPolicy](#) la Referencia AWS SDK for Java 2.x de la API.

PushImageCmd

En el siguiente ejemplo de código, se muestra cómo usar PushImageCmd.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/**
 * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
 repository.
 *
 * @param repoName the name of the ECR repository to push the image to.
 * @param imageName the name of the Docker image.
 */
public void pushDockerImage(String repoName, String imageName) {
    System.out.println("Pushing " + imageName + " to Amazon ECR will take a few
seconds.");
    CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
        .thenApply(response -> {
            String token =
response.authorizationData().get(0).authorizationToken();
            String decodedToken = new String(Base64.getDecoder().decode(token));
            String password = decodedToken.substring(4);

            DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();

```

```

        Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
        assert repoData != null;
        String registryURL = repoData.repositoryUri().split("/")[0];

        AuthConfig authConfig = new AuthConfig()
            .withUsername("AWS")
            .withPassword(password)
            .withRegistryAddress(registryURL);
        return authConfig;
    })
    .thenCompose(authConfig -> {
        DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
        Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
        getDockerClient().tagImageCmd(imageName + ":latest",
repoData.repositoryUri() + ":latest", imageName).exec();
        try {

getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
            System.out.println("The " + imageName + " was pushed to ECR");

        } catch (InterruptedException e) {
            throw (RuntimeException) e.getCause();
        }
        return CompletableFuture.completedFuture(authConfig);
    });

    authResponseFuture.join();
}

```

- Para obtener más información sobre la API, consulta [PushImageCmd](#) la Referencia AWS SDK for Java 2.x de la API.

SetRepositoryPolicy

En el siguiente ejemplo de código, se muestra cómo usar SetRepositoryPolicy.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 * @throws RepositoryPolicyNotFoundException if the repository policy does not
exist.
 * @throws EcrException if there is an unexpected error
setting the repository policy.
 */
public void setRepoPolicy(String repoName, String iamRole) {
    /*
        This example policy document grants the specified AWS principal the
permission to perform the
        `ecr:BatchGetImage` action. This policy is designed to allow the specified
principal
        to retrieve Docker images from the ECR repository.
    */
    String policyDocumentTemplate = ""
        {
            "Version" : "2012-10-17",
            "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                    "AWS" : "%s"
                },
                "Action" : "ecr:BatchGetImage"
            } ]
        }
    """;

    String policyDocument = String.format(policyDocumentTemplate, iamRole);
}
```

```
SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
    .repositoryName(repoName)
    .policyText(policyDocument)
    .build();

CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
response.whenComplete((resp, ex) -> {
    if (resp != null) {
        System.out.println("Repository policy set successfully.");
    } else {
        Throwable cause = ex.getCause();
        if (cause instanceof RepositoryPolicyNotFoundException) {
            throw (RepositoryPolicyNotFoundException) cause;
        } else if (cause instanceof EcrException) {
            throw (EcrException) cause;
        } else {
            String errorMessage = "Unexpected error: " + cause.getMessage();
            throw new RuntimeException(errorMessage, cause);
        }
    }
});
response.join();
}
```

- Para obtener más información sobre la API, consulta [SetRepositoryPolicy](#) la Referencia AWS SDK for Java 2.x de la API.

StartLifecyclePolicyPreview

En el siguiente ejemplo de código, se muestra cómo usar StartLifecyclePolicyPreview.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


```
/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag       The tag of the image to verify.
 * @throws EcrException   if there is an error retrieving the image
 information from Amazon ECR.
 * @throws CompletionException if the asynchronous operation completes
 exceptionally.
 */
public void verifyImage(String repositoryName, String imageTag) {
    DescribeImagesRequest request = DescribeImagesRequest.builder()
        .repositoryName(repositoryName)
        .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
        .build();

    CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
    response.whenComplete((describeImagesResponse, ex) -> {
        if (ex != null) {
            if (ex instanceof CompletionException) {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            } else {
                throw new RuntimeException("Unexpected error: " +
ex.getCause());
            }
        } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
            System.out.println("Image is present in the repository.");
        } else {
            System.out.println("Image is not present in the repository.");
        }
    });

    // Wait for the CompletableFuture to complete.
    response.join();
}
```

```
}
```

- Para obtener más información sobre la API, consulta [StartLifecyclePolicyPreview](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Conozca las operaciones principales de Amazon ECR

En el siguiente ejemplo de código, se muestra cómo:

- Cree un repositorio de Amazon ECR.
- Establezca políticas de repositorio.
- Recupera los URI del repositorio.
- Obtenga los tokens de autorización de Amazon ECR.
- Establezca políticas de ciclo de vida para los repositorios de Amazon ECR.
- Inserte una imagen de Docker en un repositorio de Amazon ECR.
- Compruebe la existencia de una imagen en un repositorio de Amazon ECR.
- Enumere los repositorios de Amazon ECR de su cuenta y obtenga detalles sobre ellos.
- Elimine los repositorios de Amazon ECR.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute un escenario interactivo que demuestre las características de Amazon ECR.

```
import software.amazon.awssdk.services.ecr.model.EcrException;  
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;  
  
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
 *
 * This Java code example requires an IAM Role that has permissions to interact with
 the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles\_create.html
 *
 * This Java scenario example requires a local docker image named echo-text. Without
 a local image,
 * this Java program will not successfully run. For more information including how
 to create the local
 * image, see:
 *
 * /getting\_started\_scenarios/ecr\_scenario/README
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) {
        final String usage = ""
            Usage: <iamRoleARN> <accountId>

            Where:
                iamRoleARN - The IAM role ARN that has the necessary permissions to
access and manage the Amazon ECR repository.
                accountId - Your AWS account number.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        ECRActions ecrActions = new ECRActions();
        String iamRole = args[0];
        String accountId = args[1];
    }
}
```

```
String localImageName;

Scanner scanner = new Scanner(System.in);
System.out.println("""
    The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
container registry
    service provided by AWS. It allows developers and organizations to
securely
    store, manage, and deploy Docker container images.
    ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
    from building and testing to production deployment.\s

    The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides a
set of methods to
    programmatically interact with the Amazon ECR service. This allows
developers to
    automate the storage, retrieval, and management of container images as
part of their application
    deployment pipelines. With ECR, teams can focus on building and
deploying their
    applications without having to worry about the underlying
infrastructure required to
    host and manage a container registry.

    This scenario walks you through how to perform key operations for this
service.

    Let's get started...

    You have two choices:
    1 - Run the entire program.
    2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
    this program that did not complete).
    """);

while (true) {
    String input = scanner.nextLine();
    if (input.trim().equalsIgnoreCase("1")) {
        System.out.println("Continuing with the program...");
        System.out.println("");
        break;
    } else if (input.trim().equalsIgnoreCase("2")) {
        String repoName = "echo-text";
```

```
        ecrActions.deleteECRRepository(repoName);
        return;
    } else {
        // Handle invalid input.
        System.out.println("Invalid input. Please try again.");
    }
}

waitForInputToContinue(scanner);
System.out.println(DASHES);

System.out.println("""
    1. Create an ECR repository.

    The first task is to ensure we have a local Docker image named echo-
text.

    If this image exists, then an Amazon ECR repository is created.

    An ECR repository is a private Docker container repository provided
    by Amazon Web Services (AWS). It is a managed service that makes it easy
    to store, manage, and deploy Docker container images.\s
    """);

// Ensure that a local docker image named echo-text exists.
boolean doesExist = ecrActions.isEchoTextImagePresent();
String repoName;
if (!doesExist){
    System.out.println("The local image named echo-text does not exist");
    return;
} else {
    localImageName = "echo-text";
    repoName = "echo-text";
}

try {
    String repoArn = ecrActions.createECRRepository(repoName);
    System.out.println("The ARN of the ECR repository is " + repoArn);

} catch (IllegalArgumentException e) {
    System.err.println("Invalid repository name: " + e.getMessage());
    return;
} catch (RuntimeException e) {
    System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
```

```

        e.printStackTrace();
        return;
    }
    waitForInputToContinue(scanner);

```

```

System.out.println(DASHES);
System.out.println("""
2. Set an ECR repository policy.

```

Setting an ECR repository policy using the `setRepositoryPolicy` function is crucial for maintaining the security and integrity of your container images. The repository policy allows you to define specific rules and restrictions for accessing and managing the images stored within your ECR repository.

```

""");
waitForInputToContinue(scanner);
try {
    ecrActions.setRepoPolicy(repoName, iamRole);

} catch (RepositoryPolicyNotFoundException e) {
    System.err.println("Invalid repository name: " + e.getMessage());
    return;
} catch (EcrException e) {
    System.err.println("An ECR exception occurred: " + e.getMessage());
    return;
} catch (RuntimeException e) {
    System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
    return;
}
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("""
3. Display ECR repository policy.

```

Now we will retrieve the ECR policy to ensure it was successfully set.

```

""");
waitForInputToContinue(scanner);
try {
    String policyText = ecrActions.getRepoPolicy(repoName);
    System.out.println("Policy Text:");

```

```

        System.out.println(policyText);

    } catch (EcrException e) {
        System.err.println("An ECR exception occurred: " + e.getMessage());
        return;
    } catch (RuntimeException e) {
        System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
        return;
    }

    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("""
4. Retrieve an ECR authorization token.

You need an authorization token to securely access and interact with the
Amazon ECR registry.
The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
and interacting with an Amazon ECR repository. This operation is responsible
for obtaining a
valid authorization token, which is required to authenticate your requests
to the ECR service.

Without a valid authorization token, you would not be able to perform any
operations on the
ECR repository, such as pushing, pulling, or managing your Docker images.

""");
    waitForInputToContinue(scanner);
    try {
        ecrActions.getAuthToken();

    } catch (EcrException e) {
        System.err.println("An ECR exception occurred: " + e.getMessage());
        return;
    } catch (RuntimeException e) {
        System.err.println("An error occurred while retrieving the authorization
token: " + e.getMessage());
        return;
    }
    waitForInputToContinue(scanner);

```

```

System.out.println(DASHES);
System.out.println("""
5. Get the ECR Repository URI.

```

The URI of an Amazon ECR repository is important. When you want to deploy a container image to a container orchestration platform like Amazon Elastic Kubernetes Service (EKS) or Amazon Elastic Container Service (ECS), you need to specify the full image URI, which includes the ECR repository URI. This allows the container runtime to pull the correct container image from the ECR repository.

```

""");
waitForInputToContinue(scanner);

try {
    ecrActions.getRepositoryURI(repoName);

} catch (EcrException e) {
    System.err.println("An ECR exception occurred: " + e.getMessage());
    return;

} catch (RuntimeException e) {
    System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
    return;
}
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("""
6. Set an ECR Lifecycle Policy.

```

An ECR Lifecycle Policy is used to manage the lifecycle of Docker images stored in your ECR repositories.

These policies allow you to automatically remove old or unused Docker images from your repositories, freeing up storage space and reducing costs.

This example policy helps to maintain the size and efficiency of the container registry

by automatically removing older and potentially unused images, ensuring that the storage is optimized and the registry remains up-to-date.

```

    """);
    waitForInputToContinue(scanner);
    try {
        ecrActions.setLifecyclePolicy(repoName);

    } catch (RuntimeException e) {
        System.err.println("An error occurred while setting the lifecycle
policy: " + e.getMessage());
        e.printStackTrace();
        return;
    }
    waitForInputToContinue(scanner);

```

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

```
System.out.println("""
```

7. Push a docker image to the Amazon ECR Repository.

The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR repository.

It sets up the Docker client by connecting to the local Docker host using the default port.

It then retrieves the authorization token for the ECR repository by making a call to the AWS SDK.

The method uses the authorization token to create an `AuthConfig` object, which is used to authenticate

the Docker client when pushing the image. Finally, the method tags the Docker image with the specified

repository name and image tag, and then pushes the image to the ECR repository using the Docker client.

If the push operation is successful, the method prints a message indicating that the image was pushed to ECR.

```

    """);
    waitForInputToContinue(scanner);

    try {
        ecrActions.pushDockerImage(repoName, localImageName);

    } catch (RuntimeException e) {
        System.err.println("An error occurred while pushing a local Docker image
to Amazon ECR: " + e.getMessage());

```

```

        e.printStackTrace();
        return;
    }
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("8. Verify if the image is in the ECR Repository.");
    waitForInputToContinue(scanner);
    try {
        ecrActions.verifyImage(repoName, localImageName);

    } catch (EcrException e) {
        System.err.println("An ECR exception occurred: " + e.getMessage());
        return;
    } catch (RuntimeException e) {
        System.err.println("An error occurred " + e.getMessage());
        e.printStackTrace();
        return;
    }
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("9. As an optional step, you can interact with the image
in Amazon ECR by using the CLI.");
    System.out.println("Would you like to view instructions on how to use the
CLI to run the image? (y/n)");
    String ans = scanner.nextLine().trim();
    if (ans.equalsIgnoreCase("y")) {
        String instructions = ""
            1. Authenticate with ECR - Before you can pull the image from Amazon
            ECR, you need to authenticate with the registry. You can do this using the AWS CLI:

                aws ecr get-login-password --region us-east-1 | docker login --
                username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:

                aws ecr describe-images --repository-name %s --image-ids imageTag=%s

            3. Run the Docker container and view the output using this command:

                docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
        """;
    }

```

```

        instructions = String.format(instructions, accountId, repoName,
localImageName, accountId, repoName, localImageName);
        System.out.println(instructions);
    }
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("10. Delete the ECR Repository.");
    System.out.println(
        ""
        If the repository isn't empty, you must either delete the contents of the
repository
        or use the force option (used in this scenario) to delete the repository and
have Amazon ECR delete all of its contents
        on your behalf.
        "");
    System.out.println("Would you like to delete the Amazon ECR Repository? (y/
n)");
    String delAns = scanner.nextLine().trim();
    if (delAns.equalsIgnoreCase("y")) {
        System.out.println("You selected to delete the AWS ECR resources.");

        try {
            ecrActions.deleteECRRepository(repoName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while deleting the Docker
image: " + e.getMessage());
            e.printStackTrace();
            return;
        }
    }

    System.out.println(DASHES);
    System.out.println("This concludes the Amazon ECR SDK scenario");
    System.out.println(DASHES);
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        System.out.println("");
    }
}

```

```
System.out.println("Enter 'c' followed by <ENTER> to continue:");
String input = scanner.nextLine();

if (input.trim().equalsIgnoreCase("c")) {
    System.out.println("Continuing with the program...");
    System.out.println("");
    break;
} else {
    // Handle invalid input.
    System.out.println("Invalid input. Please try again.");
}
}
}
}
```

Una clase contenedora para los métodos del SDK de Amazon ECR.

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
```

```
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
import software.amazon.awssdk.services.ecr.model.Repository;
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
import
    software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

public class ECRActions {
    private static EcrAsyncClient ecrClient;

    private static DockerClient dockerClient;

    private static Logger logger = LoggerFactory.getLogger(ECRActions.class);

    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
     string if the operation failed.
     * @throws IllegalArgumentException If repository name is invalid.
     * @throws RuntimeException if an error occurs while creating the
     repository.
     */
    public String createECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        CreateRepositoryRequest request = CreateRepositoryRequest.builder()
            .repositoryName(repoName)
            .build();
    }
}
```

```

        CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
successfully.");
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }

/**
 * Deletes an ECR (Elastic Container Registry) repository.
 *
 * @param repoName the name of the repository to delete.
 * @throws IllegalArgumentException if the repository name is null or empty.
 * @throws EcrException if there is an error deleting the repository.
 * @throws RuntimeException if an unexpected error occurs during the deletion
process.
 */
    public void deleteECRRepository(String repoName) {

```

```
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
            .force(true)
            .repositoryName(repoName)
            .build();

        CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " + repoName +
" repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }

    private static DockerClient getDockerClient() {
        String osName = System.getProperty("os.name");
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
default port.
            DockerCmdExecFactory dockerCmdExecFactory = new
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
            dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory).
```

```

    } else {
        dockerClient = DockerClientBuilder.getInstance().build();
    }
    return dockerClient;
}

/**
 * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
 *
 * @return the configured ECR asynchronous client.
 */
private static EcrAsyncClient getAsyncClient() {

    /**
     The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
    version 2,
     and it is designed to provide a high-performance, asynchronous HTTP client
    for interacting with AWS services.
     It uses the Netty framework to handle the underlying network communication
    and the Java NIO API to
     provide a non-blocking, event-driven approach to HTTP requests and
    responses.
     */
    SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
        .maxConcurrency(50) // Adjust as needed.
        .connectionTimeout(Duration.ofSeconds(60)) // Set the connection
    timeout.
        .readTimeout(Duration.ofSeconds(60)) // Set the read timeout.
        .writeTimeout(Duration.ofSeconds(60)) // Set the write timeout.
        .build();

    ClientOverrideConfiguration overrideConfig =
    ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2)) // Set the overall API call
    timeout.
        .apiCallAttemptTimeout(Duration.ofSeconds(90)) // Set the individual
    call attempt timeout.
        .build();

    if (ecrClient == null) {
        ecrClient = EcrAsyncClient.builder()
            .region(Region.US_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)

```



```
        .build();
    }
    return ecrClient;
}

/**
 * Sets the lifecycle policy for the specified repository.
 *
 * @param repoName the name of the repository for which to set the lifecycle
policy.
 */
public void setLifecyclePolicy(String repoName) {
    /**
     This policy helps to maintain the size and efficiency of the container
registry
     by automatically removing older and potentially unused images,
     ensuring that the storage is optimized and the registry remains up-to-
date.
    */
    String polText = ""
        {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14
                    },
                    "action": {
                        "type": "expire"
                    }
                }
            ]
        }
        """;

    StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
StartLifecyclePolicyPreviewRequest.builder()
    .lifecyclePolicyText(polText)
    .repositoryName(repoName)
    .build();
}
```

```

        CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
        response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
            if (lifecyclePolicyPreviewResponse != null) {
                System.out.println("Lifecycle policy preview started
successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });
        // Wait for the CompletableFuture to complete.
        response.join();
    }

/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag       The tag of the image to verify.
 * @throws EcrException   if there is an error retrieving the image
information from Amazon ECR.
 * @throws CompletionException if the asynchronous operation completes
exceptionally.
 */
public void verifyImage(String repositoryName, String imageTag) {
    DescribeImagesRequest request = DescribeImagesRequest.builder()
        .repositoryName(repositoryName)
        .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
        .build();

    CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
    response.whenComplete((describeImagesResponse, ex) -> {
        if (ex != null) {
            if (ex instanceof CompletionException) {
                Throwable cause = ex.getCause();
            }
        }
    });
}

```

```

        if (cause instanceof EcrException) {
            throw (EcrException) cause;
        } else {
            throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
        }
    } else {
        throw new RuntimeException("Unexpected error: " +
ex.getCause());
    }
} else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
    System.out.println("Image is present in the repository.");
} else {
    System.out.println("Image is not present in the repository.");
}
});

// Wait for the CompletableFuture to complete.
response.join();
}

/**
 * Retrieves the repository URI for the specified repository name.
 *
 * @param repoName the name of the repository to retrieve the URI for.
 * @return the repository URI for the specified repository name.
 * @throws EcrException if there is an error retrieving the repository
information.
 * @throws CompletionException if the asynchronous operation completes
exceptionally.
 */
public void getRepositoryURI(String repoName) {
    DescribeRepositoriesRequest request = DescribeRepositoriesRequest.builder()
        .repositoryNames(repoName)
        .build();

    CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
    response.whenComplete((describeRepositoriesResponse, ex) -> {
        if (ex != null) {
            Throwable cause = ex.getCause();
            if (cause instanceof InterruptedException) {
                Thread.currentThread().interrupt();
            }
        }
    });
}

```

```

        String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
        throw new RuntimeException(errorMessage, cause);
    } else if (cause instanceof EcrException) {
        throw (EcrException) cause;
    } else {
        String errorMessage = "Unexpected error: " + cause.getMessage();
        throw new RuntimeException(errorMessage, cause);
    }
} else {
    if (describeRepositoriesResponse != null) {
        if (!describeRepositoriesResponse.repositories().isEmpty()) {
            String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
            System.out.println("Repository URI found: " +
repositoryUri);
        } else {
            System.out.println("No repositories found for the given
name.");
        }
    } else {
        System.err.println("No response received from
describeRepositories.");
    }
}
});
response.join();
}

/**
 * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
 * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
 * If the operation is successful, the method prints the token to the console.
 * If an exception occurs, the method handles the exception and prints the error
message.
 *
 * @throws EcrException    if there is an error retrieving the authorization
token from ECR.
 * @throws RuntimeException if there is an unexpected error during the
operation.
 */
public void getAuthToken() {

```

```

        CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
        response.whenComplete((authorizationTokenResponse, ex) -> {
            if (authorizationTokenResponse != null) {
                AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
                String token = authorizationData.authorizationToken();
                if (!token.isEmpty()) {
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }

/**
 * Gets the repository policy for the specified repository.
 *
 * @param repoName the name of the repository.
 * @throws EcrException if an AWS error occurs while getting the repository
policy.
 */
    public String getRepoPolicy(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);

```

```

        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy retrieved successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }

    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole the IAM role to be granted access to the repository.
     * @throws RepositoryPolicyNotFoundException if the repository policy does not
    exist.
     * @throws EcrException if there is an unexpected error
    setting the repository policy.
     */
    public void setRepoPolicy(String repoName, String iamRole) {
        /**
         This example policy document grants the specified AWS principal the
    permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the specified
    principal
         to retrieve Docker images from the ECR repository.
         */
        String policyDocumentTemplate = ""
            {
                "Version" : "2012-10-17",
                "Statement" : [ {
                    "Sid" : "new statement",
                    "Effect" : "Allow",
                    "Principal" : {
                        "AWS" : "%s"
                    }
                }
            ]
        }
    }

```

```

        },
        "Action" : "ecr:BatchGetImage"
    } ]
}
""";

String policyDocument = String.format(policyDocumentTemplate, iamRole);
SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
    .repositoryName(repoName)
    .policyText(policyDocument)
    .build();

CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
response.whenComplete((resp, ex) -> {
    if (resp != null) {
        System.out.println("Repository policy set successfully.");
    } else {
        Throwable cause = ex.getCause();
        if (cause instanceof RepositoryPolicyNotFoundException) {
            throw (RepositoryPolicyNotFoundException) cause;
        } else if (cause instanceof EcrException) {
            throw (EcrException) cause;
        } else {
            String errorMessage = "Unexpected error: " + cause.getMessage();
            throw new RuntimeException(errorMessage, cause);
        }
    }
});
response.join();
}

/**
 * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
 *
 * @param repoName the name of the ECR repository to push the image to.
 * @param imageName the name of the Docker image.
 */
public void pushDockerImage(String repoName, String imageName) {
    System.out.println("Pushing " + imageName + " to Amazon ECR will take a few
seconds.");
}

```

```

        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
    .thenApply(response -> {
        String token =
response.authorizationData().get(0).authorizationToken();
        String decodedToken = new String(Base64.getDecoder().decode(token));
        String password = decodedToken.substring(4);

        DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
        Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
        assert repoData != null;
        String registryURL = repoData.repositoryUri().split("/")[0];

        AuthConfig authConfig = new AuthConfig()
            .withUsername("AWS")
            .withPassword(password)
            .withRegistryAddress(registryURL);
        return authConfig;
    })
    .thenCompose(authConfig -> {
        DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
        Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
        getDockerClient().tagImageCmd(imageName + ":latest",
repoData.repositoryUri() + ":latest", imageName).exec();
        try {

getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
            System.out.println("The " + imageName + " was pushed to ECR");

        } catch (InterruptedException e) {
            throw (RuntimeException) e.getCause();
        }
        return CompletableFuture.completedFuture(authConfig);
    });

    authResponseFuture.join();
}

```



```
// Make sure local image echo-text exists.
public boolean isEchoTextImagePresent() {
    try {
        List<Image> images = getDockerClient().listImagesCmd().exec();
        boolean helloWorldFound = false;
        for (Image image : images) {
            String[] repoTags = image.getRepoTags();
            if (repoTags != null) {
                for (String tag : repoTags) {
                    if (tag.startsWith("echo-text")) {
                        System.out.println(tag);
                        helloWorldFound = true;
                    }
                }
            }
        }
        if (helloWorldFound) {
            System.out.println("The local image named echo-text exists.");
            return true;
        } else {
            System.out.println("The local image named echo-text does not
exist.");
            return false;
        }
    } catch (DockerClientException ex) {
        logger.error("ERROR: " + ex.getMessage());
        return false;
    }
}
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
 - [CreateRepository](#)
 - [DeleteRepository](#)
 - [DescribeImages](#)
 - [DescribeRepositories](#)
 - [GetAuthorizationToken](#)
 - [GetRepositoryPolicy](#)

- [SetRepositoryPolicy](#)
- [StartLifecyclePolicyPreview](#)

Ejemplos de Amazon ECS usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes AWS SDK for Java 2.x mediante Amazon ECS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

CreateCluster

En el siguiente ejemplo de código, se muestra cómo usar CreateCluster.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateCluster](#) la Referencia AWS SDK for Java 2.x de la API.

CreateService

En el siguiente ejemplo de código, se muestra cómo usar CreateService.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateService](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteService

En el siguiente ejemplo de código, se muestra cómo usar DeleteService.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteService](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeClusters

En el siguiente ejemplo de código, se muestra cómo usar DescribeClusters.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeClusters](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeTasks

En el siguiente ejemplo de código, se muestra cómo usar DescribeTasks.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeTasks](#) la Referencia AWS SDK for Java 2.x de la API.

ListClusters

En el siguiente ejemplo de código, se muestra cómo usar `ListClusters`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListClusters](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateService

En el siguiente ejemplo de código, se muestra cómo usar UpdateService.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [UpdateService](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Elastic Load Balancing: versión 2 con SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x con Elastic Load Balancing, versión 2.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción al equilibrador de carga elástico

En los siguientes ejemplos de código se muestra cómo empezar a utilizar un equilibrador de carga elástico.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public class HelloLoadBalancer {  
  
    public static void main(String[] args) {  
        ElasticLoadBalancingV2Client loadBalancingV2Client =  
        ElasticLoadBalancingV2Client.builder()  
            .region(Region.US_EAST_1)
```



```
        .build();

        DescribeLoadBalancersResponse loadBalancersResponse =
loadBalancingV2Client
        .describeLoadBalancers(r -> r.pageSize(10));
        List<LoadBalancer> loadBalancerList =
loadBalancersResponse.loadBalancers();
        for (LoadBalancer lb : loadBalancerList)
            System.out.println("Load Balancer DNS name = " +
lb.dnsName());
    }
}
```

- Para obtener más información sobre la API, consulta [DescribeLoadBalancers](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateListener

En el siguiente ejemplo de código, se muestra cómo usar CreateListener.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
```

```
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();
```

```
        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

        .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

- Para obtener más información sobre la API, consulta [CreateListener](#) la Referencia AWS SDK for Java 2.x de la API.

CreateLoadBalancer

En el siguiente ejemplo de código, se muestra cómo usar CreateLoadBalancer.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/*
```

```
* Creates an Elastic Load Balancing load balancer that uses the specified
* subnets
* and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
```

```
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

- Para obtener más información sobre la API, consulta [CreateLoadBalancer](#) la Referencia AWS SDK for Java 2.x de la API.

CreateTargetGroup

En el siguiente ejemplo de código, se muestra cómo usar `CreateTargetGroup`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}
```

- Para obtener más información sobre la API, consulta [CreateTargetGroup](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteLoadBalancer

En el siguiente ejemplo de código, se muestra cómo usar DeleteLoadBalancer.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}
```

- Para obtener más información sobre la API, consulta [DeleteLoadBalancer](#) en la Referencia AWS SDK for Java 2.x de la API.

DeleteTargetGroup

En el siguiente ejemplo de código, se muestra cómo usar DeleteTargetGroup.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


```
// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}
```

- Para obtener más información sobre la API, consulta [DeleteTargetGroup](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeTargetHealth

En el siguiente ejemplo de código, se muestra cómo usar DescribeTargetHealth.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}
```

- Para obtener más información sobre la API, consulta [DescribeTargetHealth](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Creación y administración de un servicio resiliente

El siguiente ejemplo de código muestra cómo crear un servicio web con equilibrio de carga que muestre recomendaciones de libros, películas y canciones. El ejemplo muestra cómo responde el servicio a los errores y cómo reestructurarlo para aumentar la resiliencia cuando se produzcan errores.

- Utilice un grupo de Amazon EC2 Auto Scaling para crear instancias de Amazon Elastic Compute Cloud (Amazon EC2) basadas en una plantilla de lanzamiento y para mantener el número de instancias dentro de un rango específico.
- Administre y distribuya las solicitudes HTTP con Elastic Load Balancing.
- Supervise el estado de las instancias de un grupo de escalado automático y reenvíe las solicitudes solo a las instancias en buen estado.
- Ejecute un servidor web Python en cada instancia de EC2 para administrar las solicitudes HTTP. El servidor web responde con recomendaciones y comprobaciones de estado.
- Simule un servicio de recomendaciones con una tabla de Amazon DynamoDB.
- Controle la respuesta del servidor web a las solicitudes y las comprobaciones de estado actualizando AWS Systems Manager los parámetros.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute el escenario interactivo en un símbolo del sistema.

```
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 =
HttpClientClients.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);
}
}

```

Cree una clase que agrupe las acciones de escalado automático y 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.");
    }
}

```

Cree una clase que resuma las acciones de 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 "";
}
}

```

Cree una clase que utilice DynamoDB para simular un servicio de recomendaciones.

```

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);
}
}

```

Cree una clase que agrupe las acciones de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de 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](#)

MediaStore ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with MediaStore.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

CreateContainer

En el siguiente ejemplo de código, se muestra cómo usar `CreateContainer`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateContainer](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteContainer

En el siguiente ejemplo de código, se muestra cómo usar DeleteContainer.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteContainer](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteObject

En el siguiente ejemplo de código, se muestra cómo usar DeleteObject.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteObject](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeContainer

En el siguiente ejemplo de código, se muestra cómo usar DescribeContainer.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeContainer](#) la Referencia AWS SDK for Java 2.x de la API.

GetObject

En el siguiente ejemplo de código, se muestra cómo usar GetObject.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetObject](#) la Referencia AWS SDK for Java 2.x de la API.

ListContainers

En el siguiente ejemplo de código, se muestra cómo usar ListContainers.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListContainers](#) la Referencia AWS SDK for Java 2.x de la API.

PutObject

En el siguiente ejemplo de código, se muestra cómo usar PutObject.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [PutObject](#) la Referencia AWS SDK for Java 2.x de la API.

OpenSearch Ejemplos de servicios que utilizan SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante el uso del OpenSearch servicio AWS SDK for Java 2.x with.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

CreateDomain

En el siguiente ejemplo de código, se muestra cómo usar CreateDomain.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateDomain](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteDomain

En el siguiente ejemplo de código, se muestra cómo usar DeleteDomain.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteDomain](#) la Referencia AWS SDK for Java 2.x de la API.

ListDomainNames

En el siguiente ejemplo de código, se muestra cómo usar ListDomainNames.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListDomainNames](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateDomainConfig

En el siguiente ejemplo de código, se muestra cómo usar UpdateDomainConfig.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [UpdateDomainConfig](#) la Referencia AWS SDK for Java 2.x de la API.

EventBridge ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with EventBridge.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

¿Hola EventBridge

Los siguientes ejemplos de código muestran cómo empezar a usarlo EventBridge.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
*/
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
            .region(region)
            .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 obtener más información sobre la API, consulta [ListEventBuses](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

DeleteRule

En el siguiente ejemplo de código, se muestra cómo usar DeleteRule.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
}
```

- Para obtener más información sobre la API, consulta [DeleteRule](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeRule

En el siguiente ejemplo de código, se muestra cómo usar DescribeRule.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    try {
        DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
            .name(eventRuleName)
            .build();

        DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
        System.out.println("The state of the rule is " +
response.stateAsString());

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

- Para obtener más información sobre la API, consulta [DescribeRule](#) la Referencia AWS SDK for Java 2.x de la API.

DisableRule

En el siguiente ejemplo de código, se muestra cómo usar `DisableRule`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Deshabilitar una regla por su nombre de regla.

```
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 obtener más información sobre la API, consulta [DisableRule](#) la Referencia AWS SDK for Java 2.x de la API.

EnableRule

En el siguiente ejemplo de código, se muestra cómo usar `EnableRule`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Habilitar una regla por su nombre de regla.

```
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 obtener más información sobre la API, consulta [EnableRule](#) la Referencia AWS SDK for Java 2.x de la API.

ListRuleNamesByTarget

En el siguiente ejemplo de código, se muestra cómo usar `ListRuleNamesByTarget`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Enumerar todos los nombres de regla por el 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 obtener más información sobre la API, consulta [ListRuleNamesByTarget](#) la Referencia AWS SDK for Java 2.x de la API.

ListRules

En el siguiente ejemplo de código, se muestra cómo usar ListRules.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Habilitar una regla por su nombre de regla.

```
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 obtener más información sobre la API, consulta [ListRules](#) la Referencia AWS SDK for Java 2.x de la API.

ListTargetsByRule

En el siguiente ejemplo de código, se muestra cómo usar `ListTargetsByRule`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Enumerar todos los destinos para una regla por el nombre de regla.

```
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 obtener más información sobre la API, consulta [ListTargetsByRule](#) la Referencia AWS SDK for Java 2.x de la API.

PutEvents

En el siguiente ejemplo de código, se muestra cómo usar PutEvents.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
    String json = "{" +
        "\"UserEmail\": \"" + email + "\", " +
        "\"Message\": \"This event was generated by example code.\", " +
        "\"UtcTime\": \"Now.\" " +
        "}";

    PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
        .source("ExampleSource")
        .detail(json)
        .detailType("ExampleType")
        .build();

    PutEventsRequest eventsRequest = PutEventsRequest.builder()
        .entries(entry)
        .build();

    eventBrClient.putEvents(eventsRequest);
}
```

- Para obtener más información sobre la API, consulta [PutEvents](#) la Referencia AWS SDK for Java 2.x de la API.

PutRule

En el siguiente ejemplo de código, se muestra cómo usar PutRule.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear una regla programada.

```
public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {
    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .scheduleExpression(cronExpression)
            .state("ENABLED")
            .description("A test rule that runs on a schedule created by the
Java API")
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

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

Crear una regla que se active cuando se agregue un objeto a un bucket de Amazon Simple Storage Service.

```
// Create a new event rule that triggers when an Amazon S3 object is created in
```

```

// a bucket.
public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
    String eventRuleName) {
    String pattern = "{\n" +
        "  \"source\": [\"aws.s3\"],\n" +
        "  \"detail-type\": [\"Object Created\"],\n" +
        "  \"detail\": {\n" +
        "    \"bucket\": {\n" +
        "      \"name\": [\"\" + bucketName + "\"]\n" +
        "    }\n" +
        "  }\n" +
        "}";

    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .description("Created by using the AWS SDK for Java v2")
            .name(eventRuleName)
            .eventPattern(pattern)
            .roleArn(roleArn)
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

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


```

- Para obtener más información sobre la API, consulta [PutRule](#) la Referencia AWS SDK for Java 2.x de la API.

PutTargets

En el siguiente ejemplo de código, se muestra cómo usar PutTargets.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Agregue un tema de Amazon SNS como destino de una regla.

```
// 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 + ".");
}
```

Agregue un transformador de entrada al destino de una regla.

```
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 obtener más información sobre la API, consulta [PutTargets](#) la Referencia AWS SDK for Java 2.x de la API.

RemoveTargets

En el siguiente ejemplo de código, se muestra cómo usar RemoveTargets.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Eliminar todos los destinos de una regla por el nombre de regla.

```
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 obtener más información sobre la API, consulta [RemoveTargets](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Introducción a las reglas y los destinos

En el siguiente ejemplo de código, se muestra cómo:

- Crear una regla y agregarle un destino
- Habilitar y deshabilitar reglas.
- Enumerar y actualizar reglas y destinos.
- Enviar eventos y, después, limpiar los recursos

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 putOb = 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [DeleteRule](#)
 - [DescribeRule](#)
 - [DisableRule](#)
 - [EnableRule](#)
 - [ListRuleNamesByTarget](#)
 - [ListRules](#)
 - [ListTargetsByRule](#)
 - [PutEvents](#)
 - [PutRule](#)
 - [PutTargets](#)

Ejemplos de pronóstico usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Java 2.x with Forecast.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

CreateDataset

En el siguiente ejemplo de código, se muestra cómo usar CreateDataset.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateDataset](#) la Referencia AWS SDK for Java 2.x de la API.

CreateForecast

En el siguiente ejemplo de código, se muestra cómo usar CreateForecast.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


```
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 obtener más información sobre la API, consulta [CreateForecast](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteDataset

En el siguiente ejemplo de código, se muestra cómo usar DeleteDataset.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteDataset](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteForecast

En el siguiente ejemplo de código, se muestra cómo usar DeleteForecast.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteForecast](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeForecast

En el siguiente ejemplo de código, se muestra cómo usar DescribeForecast.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeForecast](#) la Referencia AWS SDK for Java 2.x de la API.

ListDatasetGroups

En el siguiente ejemplo de código, se muestra cómo usar ListDatasetGroups.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListDatasetGroups](#) la Referencia AWS SDK for Java 2.x de la API.

ListForecasts

En el siguiente ejemplo de código, se muestra cómo usar `ListForecasts`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListForecasts](#) la Referencia AWS SDK for Java 2.x de la API.

AWS Glue ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with AWS Glue.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

¿Hola AWS Glue

En los siguientes ejemplos de código se muestra cómo empezar a utilizar AWS Glue.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListJobs](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateCrawler

En el siguiente ejemplo de código, se muestra cómo usar `CreateCrawler`.

SDK para Java 2.x

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateCrawler](#) la Referencia AWS SDK for Java 2.x de la API.

GetCrawler

En el siguiente ejemplo de código, se muestra cómo usar `GetCrawler`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetCrawler](#) la Referencia AWS SDK for Java 2.x de la API.

GetDatabase

En el siguiente ejemplo de código, se muestra cómo usar GetDatabase.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetDatabase](#) la Referencia AWS SDK for Java 2.x de la API.

GetTables

En el siguiente ejemplo de código, se muestra cómo usar `GetTables`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetTables](#) la Referencia AWS SDK for Java 2.x de la API.

StartCrawler

En el siguiente ejemplo de código, se muestra cómo usar `StartCrawler`.

SDK para Java 2.x

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [StartCrawler](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Comenzar a ejecutar rastreadores y trabajos

En el siguiente ejemplo de código, se muestra cómo:

- Crear un rastreador que rastree un bucket de Amazon S3 público y generar una base de datos de metadatos con formato CSV.
- Enumera información sobre bases de datos y tablas en tu AWS Glue Data Catalog.
- Crear un trabajo para extraer datos CSV del bucket de S3, transformar los datos y cargar el resultado con formato JSON en otro bucket de S3.
- Incluir información sobre las ejecuciones de trabajos, ver algunos de los datos transformados y limpiar los recursos.

Para obtener más información, consulte el [tutorial: Primeros pasos con AWS Glue Studio](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateCrawler](#)
 - [CreateJob](#)
 - [DeleteCrawler](#)
 - [DeleteDatabase](#)
 - [DeleteJob](#)
 - [DeleteTable](#)
 - [GetCrawler](#)
 - [GetDatabase](#)
 - [GetDatabases](#)
 - [GetJob](#)
 - [GetJobRun](#)
 - [GetJobRuns](#)
 - [GetTables](#)
 - [ListJobs](#)
 - [StartCrawler](#)

- [StartJobRun](#)

HealthImaging ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with HealthImaging.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CopyImageSet

En el siguiente ejemplo de código, se muestra cómo usar CopyImageSet.

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 obtener más información sobre la API, consulta [CopyImageSet](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

CreateDatastore

En el siguiente ejemplo de código, se muestra cómo usar `CreateDatastore`.

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 obtener más información sobre la API, consulta [CreateDatastore](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

DeleteDatastore

En el siguiente ejemplo de código, se muestra cómo usar `DeleteDatastore`.

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 obtener más información sobre la API, consulta [DeleteDatastore](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

DeleteImageSet

En el siguiente ejemplo de código, se muestra cómo usar DeleteImageSet.

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 obtener más información sobre la API, consulta [DeleteImageSet](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

GetDICOMImportJob

En el siguiente ejemplo de código, se muestra cómo usar GetDICOMImportJob.

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 obtener más información sobre la API, consulta [getDICom ImportJob](#) en la referencia de la AWS SDK for Java 2.x API.

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

GetDatastore


En el siguiente ejemplo de código, se muestra cómo usar GetDatastore.

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 obtener más información sobre la API, consulta [GetDatastore](#) la Referencia AWS SDK for Java 2.x de la API.

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

GetImageFrame

En el siguiente ejemplo de código, se muestra cómo usar GetImageFrame.

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 obtener más información sobre la API, consulta [GetImageFrame](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

GetImageSet

En el siguiente ejemplo de código, se muestra cómo usar GetImageSet.

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 obtener más información sobre la API, consulta [GetImageSet](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

GetImageSetMetadata

En el siguiente ejemplo de código, se muestra cómo usar `GetImageSetMetadata`.

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 obtener más información sobre la API, consulta [GetImageSetMetadata](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


ListDICOMImportJobs

En el siguiente ejemplo de código, se muestra cómo usar `ListDICOMImportJobs`.

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 obtener detalles sobre la API, consulta [ListDicom ImportJobs](#) en la referencia de la AWS SDK for Java 2.x API.

 Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

ListDatastores

En el siguiente ejemplo de código, se muestra cómo usar ListDatastores.

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 obtener más información sobre la API, consulta [ListDatastores](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

ListImageSetVersions

En el siguiente ejemplo de código, se muestra cómo usar `ListImageSetVersions`.

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 obtener más información sobre la API, consulta [ListImageSetVersions](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

ListTagsForResource

En el siguiente ejemplo de código, se muestra cómo usar `ListTagsForResource`.

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 obtener más información sobre la API, consulta [ListTagsForResource](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

SearchImageSets

En el siguiente ejemplo de código, se muestra cómo usar SearchImageSets.

SDK para Java 2.x

La función de utilidad para buscar conjuntos de imágenes.

```
public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
    MedicalImagingClient medicalImagingClient,
    String datastoreId, SearchCriteria searchCriteria) {
    try {
        SearchImageSetsRequest dataStoreRequest =
SearchImageSetsRequest.builder()
            .datastoreId(datastoreId)
            .searchCriteria(searchCriteria)
            .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 núm. 1: operador IGUAL.

```
List<SearchFilter> searchFilters =
Collections.singletonList(SearchFilter.builder()
    .operator(Operator.EQUAL)
    .values(SearchByAttributeValue.builder()
        .dicomPatientId(patientId)
        .build())
)
```



```

        .build());

    SearchCriteria searchCriteria = SearchCriteria.builder()
        .filters(searchFilters)
        .build();

    List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
searchMedicalImagingImageSets(
        medicalImagingClient,
        datastoreId, searchCriteria);
    if (imageSetsMetadataSummaries != null) {
        System.out.println("The image sets for patient " + patientId + " are:\n"
            + imageSetsMetadataSummaries);
        System.out.println();
    }
}

```

Caso de uso #2: El operador BETWEEN utiliza DICOM StudyDate y 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());

    searchCriteria = SearchCriteria.builder()
        .filters(searchFilters)
        .build();

```

```

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                                datastoreId, searchCriteria);
        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 núm. 3: el operador ENTRE usa CreatedAt. Los estudios de tiempo se habían mantenido previamente.

```

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());

searchCriteria = SearchCriteria.builder()
    .filters(searchFilters)
    .build();
imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                                datastoreId, searchCriteria);
if (imageSetsMetadataSummaries != null) {
    System.out.println("The image sets searched with BETWEEN operator using
createdAt are:\n "
        + imageSetsMetadataSummaries);
    System.out.println();
}

```

Caso de uso #4: operador EQUAL en DICOM SeriesInstance UID y BETWEEN en UpdatedAt y ordena la respuesta en orden ASC en el campo UpdatedAt.

```
Instant startDate = Instant.parse("1985-04-12T23:20:50.52Z");
Instant endDate = Instant.now();

searchFilters = Arrays.asList(
    SearchFilter.builder()
        .operator(Operator.EQUAL)
        .values(SearchByAttributeValue.builder()
            .dicomSeriesInstanceUID(seriesInstanceUID)
            .build())
        .build(),
    SearchFilter.builder()
        .operator(Operator.BETWEEN)
        .values(
            SearchByAttributeValue.builder().updatedAt(startDate).build(),
            SearchByAttributeValue.builder().updatedAt(endDate).build()
        ).build());

Sort sort =
Sort.builder().sortOrder(SortOrder.ASC).sortField(SortField.UPDATED_AT).build();

searchCriteria = SearchCriteria.builder()
    .filters(searchFilters)
    .sort(sort)
    .build();

imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
    datastoreId, searchCriteria);
if (imageSetsMetadataSummaries != null) {
    System.out.println("The image sets searched with EQUAL operator on
DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response\n" +
        "in ASC order on updatedAt field are:\n "
        + imageSetsMetadataSummaries);
    System.out.println();
}
```

- Para obtener más información sobre la API, consulte la Referencia de la API. [SearchImageSets](#) AWS SDK for Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

StartDICOMImportJob

En el siguiente ejemplo de código, se muestra cómo usar `StartDICOMImportJob`.


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 obtener más información sobre la API, consulte [StartDicom ImportJob](#) en la referencia de la AWS SDK for Java 2.x API.

 Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

TagResource

En el siguiente ejemplo de código, se muestra cómo usar TagResource.

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 obtener más información sobre la API, consulta [TagResource](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

UntagResource

En el siguiente ejemplo de código, se muestra cómo usar UntagResource.

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 obtener más información sobre la API, consulta [UntagResource](#) la Referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

UpdateImageSetMetadata

En el siguiente ejemplo de código, se muestra cómo usar UpdateImageSetMetadata.

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();

        UpdateImageSetMetadataResponse response =
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

        System.out.println("The image set metadata was updated" + response);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Caso de uso #1: Inserta o actualiza un atributo.

```
final String insertAttributes = ""
{
    "SchemaVersion": 1.1,
    "Study": {
        "DICOM": {
            "StudyDescription": "CT CHEST"
        }
    }
}
```

```

        }
    }
    """;

    MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
        .dicomUpdates(DICOMUpdates.builder()
            .updateableAttributes(SdkBytes.fromByteBuffer(
                ByteBuffer.wrap(insertAttributes
                    .getBytes(StandardCharsets.UTF_8))))
            .build())
        .build();

    updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
        imagesetId,
        versionid, metadataInsertUpdates);

```

Caso de uso #2: eliminar un atributo.

```

    final String removeAttributes = ""
        {
            "SchemaVersion": 1.1,
            "Study": {
                "DICOM": {
                    "StudyDescription": "CT CHEST"
                }
            }
        }
    """;

    MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
        .dicomUpdates(DICOMUpdates.builder()
            .removableAttributes(SdkBytes.fromByteBuffer(
                ByteBuffer.wrap(removeAttributes
                    .getBytes(StandardCharsets.UTF_8))))
            .build())
        .build();

    updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
        imagesetId,
        versionid, metadataRemoveUpdates);

```

Caso de uso #3: eliminar una instancia.


```

final String removeInstance = ""
    {
        "SchemaVersion": 1.1,
        "Study": {
            "Series": {
                "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                    "Instances": {
                        "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1":
                    {}
                }
            }
        }
    }
};

MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
    .dicomUpdates(DICOMUpdates.builder()
        .removableAttributes(SdkBytes.fromByteBuffer(
            ByteBuffer.wrap(removeInstance
                .getBytes(StandardCharsets.UTF_8))))
        .build())
    .build();

updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
    imagesetId,
        versionid, metadataRemoveUpdates);

```

- Para obtener más información sobre la API, consulte [UpdateImageSetMetadata](#) la referencia AWS SDK for Java 2.x de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Escenarios

Etiquetar un almacén de datos

El siguiente ejemplo de código muestra cómo etiquetar un banco HealthImaging de datos.

SDK para Java 2.x

Para etiquetar un almacén de datos

```
final String dataStoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

TagResource.tagMedicalImagingResource(medicalImagingClient,
dataStoreArn,
ImmutableMap.of("Deployment", "Development"));
```

Función de utilidad para etiquetar un 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 enumerar las etiquetas de almacenes de datos

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

La función de utilidad para enumerar las etiquetas de un 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 desetiquetar un almacén de datos

```
        final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

        UntagResource.untagMedicalImagingResource(medicalImagingClient,
datastoreArn,
            Collections.singletonList("Deployment"));
    }
```

La función de utilidad para eliminar la etiqueta de un 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [ListTagsForResource](#)
 - [TagResource](#)
 - [UntagResource](#)

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Etiquetar un conjunto de imágenes

El siguiente ejemplo de código muestra cómo etiquetar un conjunto HealthImaging de imágenes.

SDK para Java 2.x

Para etiquetar un conjunto de imágenes

```
final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
```

```
TagResource.tagMedicalImagingResource(medicalImagingClient,  
imageSetArn,  
ImmutableMap.of("Deployment", "Development"));
```

Función de utilidad para etiquetar un 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 enumerar las etiquetas de un conjunto de imágenes

```
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());  
}
```

La función de utilidad para enumerar las etiquetas de un 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 desetiquetar un conjunto de imágenes

```

        final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

        UntagResource.untagMedicalImagingResource(medicalImagingClient,
            imageSetArn,
                Collections.singletonList("Deployment"));

```

La función de utilidad para eliminar la etiqueta de un 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [ListTagsForResource](#)
 - [TagResource](#)
 - [UntagResource](#)

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejemplos de IAM usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Java 2.x IAM.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a IAM

En los siguientes ejemplos de código se muestra cómo empezar a utilizar IAM.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListPolicies](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

AttachRolePolicy

En el siguiente ejemplo de código, se muestra cómo usar AttachRolePolicy.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [AttachRolePolicy](#) la Referencia AWS SDK for Java 2.x de la API.

CreateAccessKey

En el siguiente ejemplo de código, se muestra cómo usar CreateAccessKey.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateAccessKey](#) la Referencia AWS SDK for Java 2.x de la API.

CreateAccountAlias

En el siguiente ejemplo de código, se muestra cómo usar CreateAccountAlias.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateAccountAlias](#) la Referencia AWS SDK for Java 2.x de la API.

CreatePolicy

En el siguiente ejemplo de código, se muestra cómo usar CreatePolicy.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreatePolicy](#) la Referencia AWS SDK for Java 2.x de la API.

CreateRole

En el siguiente ejemplo de código, se muestra cómo usar CreateRole.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateRole](#) la Referencia AWS SDK for Java 2.x de la API.

CreateUser

En el siguiente ejemplo de código, se muestra cómo usar CreateUser.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateUser](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteAccessKey

En el siguiente ejemplo de código, se muestra cómo usar DeleteAccessKey.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteAccessKey](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteAccountAlias

En el siguiente ejemplo de código, se muestra cómo usar DeleteAccountAlias.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteAccountAlias](#) la Referencia AWS SDK for Java 2.x de la API.

DeletePolicy

En el siguiente ejemplo de código, se muestra cómo usar DeletePolicy.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeletePolicy](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteUser

En el siguiente ejemplo de código, se muestra cómo usar DeleteUser.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteUser](#) la Referencia AWS SDK for Java 2.x de la API.

DetachRolePolicy

En el siguiente ejemplo de código, se muestra cómo usar DetachRolePolicy.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DetachRolePolicy](#) la Referencia AWS SDK for Java 2.x de la API.

ListAccessKeys

En el siguiente ejemplo de código, se muestra cómo usar ListAccessKeys.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListAccessKeys](#) la Referencia AWS SDK for Java 2.x de la API.

ListAccountAliases

En el siguiente ejemplo de código, se muestra cómo usar `ListAccountAliases`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListAccountAliases](#) la Referencia AWS SDK for Java 2.x de la API.

ListUsers

En el siguiente ejemplo de código, se muestra cómo usar ListUsers.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListUsers](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateAccessKey

En el siguiente ejemplo de código, se muestra cómo usar UpdateAccessKey.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [UpdateAccessKey](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateUser

En el siguiente ejemplo de código, se muestra cómo usar `UpdateUser`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [UpdateUser](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Creación y administración de un servicio resiliente

El siguiente ejemplo de código muestra cómo crear un servicio web con equilibrio de carga que muestre recomendaciones de libros, películas y canciones. El ejemplo muestra cómo responde el servicio a los errores y cómo reestructurarlo para aumentar la resiliencia cuando se produzcan errores.

- Utilice un grupo de Amazon EC2 Auto Scaling para crear instancias de Amazon Elastic Compute Cloud (Amazon EC2) basadas en una plantilla de lanzamiento y para mantener el número de instancias dentro de un rango específico.
- Administre y distribuya las solicitudes HTTP con Elastic Load Balancing.
- Supervise el estado de las instancias de un grupo de escalado automático y reenvíe las solicitudes solo a las instancias en buen estado.
- Ejecute un servidor web Python en cada instancia de EC2 para administrar las solicitudes HTTP. El servidor web responde con recomendaciones y comprobaciones de estado.
- Simule un servicio de recomendaciones con una tabla de Amazon DynamoDB.
- Controle la respuesta del servidor web a las solicitudes y las comprobaciones de estado actualizando AWS Systems Manager los parámetros.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute el escenario interactivo en un símbolo del sistema.

```
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);
}

```

```
}  
}
```

Cree una clase que agrupe las acciones de escalado automático y 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.");
    }
}

```

Cree una clase que resuma las acciones de 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 "";
}
}

```

Cree una clase que utilice DynamoDB para simular un servicio de recomendaciones.

```

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);
}
}

```

Cree una clase que agrupe las acciones de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de 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](#)

Crear un usuario y asumir un rol

En el siguiente ejemplo de código, se muestra cómo crear un usuario y asumir un rol.

 Warning

Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como [AWS IAM Identity Center](#).

- Crear un usuario que no tenga permisos.
- Crear un rol que conceda permiso para enumerar los buckets de Amazon S3 para la cuenta.
- Agregar una política para que el usuario asuma el rol.

- Asumir el rol y enumerar los buckets de S3 con credenciales temporales, y después limpiar los recursos.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree funciones que encapsulen las acciones del usuario de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)
 - [DeletePolicy](#)
 - [DeleteRole](#)
 - [DeleteUser](#)
 - [DeleteUserPolicy](#)
 - [DetachRolePolicy](#)
 - [PutUserPolicy](#)

Trabajar con la API del creador de políticas de IAM

En el siguiente ejemplo de código, se muestra cómo:

- Cree políticas de IAM mediante la API orientada a objetos.
- Utilice la API del creador de políticas de IAM con el servicio de IAM.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

En los ejemplos se utilizan las siguientes importaciones.

```
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;
```

Cree una política basada en el tiempo.

```
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());
}

```

Cree una política con varias condiciones.

```

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());
}

```

Utilice las entidades principales en una 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::4444455556666:user/user-name"))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Permitir el acceso entre cuentas de.

```

public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b

```

```

        "111122223333")
        .effect(IamEffect.ALLOW)
        .addPrincipal(IamPrincipalType.AWS,
        EXAMPLE-BUCKET/*")
        .addAction("s3:PutObject")
        .addResource("arn:aws:s3:::DOC-
        .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());
}

```

Cree y cargue una `IamPolicy`.

```

    public String createAndUploadPolicyExample(IamClient iam, String accountID,
    String policyName) {
        // Build the policy.
        IamPolicy policy = IamPolicy.builder() // 'version' defaults to
        "2012-10-17".
        .addStatement(IamStatement.builder()
        .effect(IamEffect.ALLOW)
        .addAction("dynamodb:PutItem")
        .addResource("arn:aws:dynamodb:us-
        east-1:" + accountID
        + ":table/
        exampleTableName")
        .build())
        .build();
        // Upload the policy.
        iam.createPolicy(r ->
        r.policyName(policyName).policyDocument(policy.toJson()));
        return
        policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
    }
}

```

Descargue y trabaje con una `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 obtener información, consulte la [Guía para desarrolladores de AWS SDK for Java 2.x](#).
- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
 - [CreatePolicy](#)
 - [GetPolicy](#)
 - [GetPolicyVersion](#)

AWS IoT ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with AWS IoT.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

¿Hola AWS IoT

En los siguientes ejemplos de código se muestra cómo empezar a utilizar AWS Glue.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener información sobre la API, consulte [listThings](#) en la Referencia de la API de AWS SDK for Java 2.x .

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

AttachThingPrincipal

En el siguiente ejemplo de código, se muestra cómo usar `AttachThingPrincipal`.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [AttachThingPrincipal](#) la Referencia AWS SDK for Java 2.x de la API.

CreateKeysAndCertificate

En el siguiente ejemplo de código, se muestra cómo usar `CreateKeysAndCertificate`.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateKeysAndCertificate](#) la Referencia AWS SDK for Java 2.x de la API.

CreateThing

En el siguiente ejemplo de código, se muestra cómo usar `CreateThing`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateThing](#) la Referencia AWS SDK for Java 2.x de la API.

CreateTopicRule

En el siguiente ejemplo de código, se muestra cómo usar CreateTopicRule.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


```
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 obtener más información sobre la API, consulta [CreateTopicRule](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteCertificate

En el siguiente ejemplo de código, se muestra cómo usar DeleteCertificate.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteCertificate](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteThing

En el siguiente ejemplo de código, se muestra cómo usar DeleteThing.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteThing](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeEndpoint

En el siguiente ejemplo de código, se muestra cómo usar DescribeEndpoint.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeEndpoint](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeThing

En el siguiente ejemplo de código, se muestra cómo usar DescribeThing.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeThing](#) la Referencia AWS SDK for Java 2.x de la API.

DetachThingPrincipal

En el siguiente ejemplo de código, se muestra cómo usar `DetachThingPrincipal`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DetachThingPrincipalla Referencia AWS SDK for Java 2.x de la API](#).

ListCertificates

En el siguiente ejemplo de código, se muestra cómo usar ListCertificates.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListCertificates](#) la Referencia AWS SDK for Java 2.x de la API.

SearchIndex

En el siguiente ejemplo de código, se muestra cómo usar SearchIndex.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [SearchIndex](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateThing

En el siguiente ejemplo de código, se muestra cómo usar UpdateThing.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [UpdateThing](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Trabaje con casos de uso de la administración de dispositivos

En el siguiente ejemplo de código se muestra cómo trabajar con los casos de uso de la administración de AWS IoT dispositivos mediante el AWS IoT SDK

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with AWS IoT data.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

GetThingShadow

En el siguiente ejemplo de código, se muestra cómo usar GetThingShadow.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetThingShadow](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateThingShadow

En el siguiente ejemplo de código, se muestra cómo usar UpdateThingShadow.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [UpdateThingShadow](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Amazon Keyspaces usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Amazon Keyspaces.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Amazon Keyspaces

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon Keyspaces.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListKeyspaces](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateKeyspace

En el siguiente ejemplo de código, se muestra cómo usar CreateKeyspace.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

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

- Para obtener más información sobre la API, consulta [CreateKeyspace](#) la Referencia AWS SDK for Java 2.x de la API.

CreateTable

En el siguiente ejemplo de código, se muestra cómo usar CreateTable.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();

        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
            .name("release_date")
            .type("timestamp")
            .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
            .name("plot")
            .type("text")
            .build();

        List<ColumnDefinition> collist = new ArrayList<>();
        collist.add(defTitle);
        collist.add(defYear);
        collist.add(defReleaseDate);
        collist.add(defPlot);

        // Set the keys.
        PartitionKey yearKey = PartitionKey.builder()
            .name("year")
            .build();
```

```
PartitionKey titleKey = PartitionKey.builder()
    .name("title")
    .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(colList)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
    .build();

CreateTableResponse response = keyClient.createTable(tableRequest);
System.out.println("The table ARN is " + response.resourceArn());

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

- Para obtener más información sobre la API, consulta [CreateTable](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteKeyspace

En el siguiente ejemplo de código, se muestra cómo usar DeleteKeyspace.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        keyClient.deleteKeyspace(deleteKeyspaceRequest);


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

- Para obtener más información sobre la API, consulta [DeleteKeyspace](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteTable

En el siguiente ejemplo de código, se muestra cómo usar DeleteTable.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        keyClient.deleteTable(tableRequest);

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

- Para obtener más información sobre la API, consulta [DeleteTable](#) la Referencia AWS SDK for Java 2.x de la API.

GetKeyspace

En el siguiente ejemplo de código, se muestra cómo usar GetKeyspace.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
    }
}
```



```
        System.out.println("The " + name + " KeySpace is ready");
    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [GetKeyspace](#) la Referencia AWS SDK for Java 2.x de la API.

GetTable

En el siguiente ejemplo de código, se muestra cómo usar `GetTable`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);
        }
    }
}
```

```
        if (status.compareTo("ACTIVE") == 0) {
            tableStatus = true;
        }
        Thread.sleep(500);
    }

    List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
    for (ColumnDefinition def : cols) {
        System.out.println("The column name is " + def.name());
        System.out.println("The column type is " + def.type());
    }

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

- Para obtener más información sobre la API, consulta [GetTable](#) la Referencia AWS SDK for Java 2.x de la API.

ListKeyspaces

En el siguiente ejemplo de código, se muestra cómo usar `ListKeyspaces`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();
```

```

        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

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

```

- Para obtener más información sobre la API, consulta [ListKeyspaces](#) la Referencia AWS SDK for Java 2.x de la API.

ListTables

En el siguiente ejemplo de código, se muestra cómo usar `ListTables`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
        listRes.stream()
            .flatMap(r -> r.tables().stream())
            .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +

```

```

        " Table name: " + content.tableName()));

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

```

- Para obtener más información sobre la API, consulta [ListTables](#) la Referencia AWS SDK for Java 2.x de la API.

RestoreTable

En el siguiente ejemplo de código, se muestra cómo usar `RestoreTable`.

SDK para Java 2.x

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
    ZonedDateTime utc) {
    try {
        Instant myTime = utc.toInstant();
        RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
            .restoreTimestamp(myTime)
            .sourceTableName("Movie")
            .targetKeyspaceName(keyspaceName)
            .targetTableName("MovieRestore")
            .sourceKeyspaceName(keyspaceName)
            .build();

        RestoreTableResponse response =
            keyClient.restoreTable(restoreTableRequest);
        System.out.println("The ARN of the restored table is " +
            response.restoredTableARN());
    }
}

```

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

- Para obtener más información sobre la API, consulta [RestoreTable](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateTable

En el siguiente ejemplo de código, se muestra cómo usar UpdateTable.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void updateTable(KeyspacesClient keyClient, String keySpace,  
String tableName) {  
    try {  
        ColumnDefinition def = ColumnDefinition.builder()  
            .name("watched")  
            .type("boolean")  
            .build();  
  
        UpdateTableRequest tableRequest = UpdateTableRequest.builder()  
            .keySpaceName(keySpace)  
            .tableName(tableName)  
            .addColumnns(def)  
            .build();  
  
        keyClient.updateTable(tableRequest);  
  
    } catch (KeyspacesException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
}  
}
```

- Para obtener más información sobre la API, consulta [UpdateTable](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Introducción a los espacios de claves y las tablas

En el siguiente ejemplo de código, se muestra cómo:

- Crear un espacio de claves y una tabla. El esquema de la tabla contiene los datos de las películas y tiene habilitada point-in-time la recuperación.
- Conectarse al espacio de claves mediante una conexión TLS segura con autenticación SigV4.
- Consultar la tabla. Agregar, recuperar y actualizar datos de películas.
- Actualizar la tabla. Añadir una columna para llevar un seguimiento de las películas vistas.
- Restaurar la tabla a su estado anterior y limpiar los recursos.

SDK para Java 2.x

Note

Hay más en marcha GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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> collist = new ArrayList<>();
        collist.add(defTitle);
        collist.add(defYear);
        collist.add(defReleaseDate);
        collist.add(defPlot);
    }
}
```

```
// Set the keys.
PartitionKey yearKey = PartitionKey.builder()
    .name("year")
    .build();

PartitionKey titleKey = PartitionKey.builder()
    .name("title")
    .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(colList)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
    .build();

CreateTableResponse response = keyClient.createTable(tableRequest);
System.out.println("The table ARN is " + response.resourceArn());

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

}

public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();
```

```
        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

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

public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
        System.out.println("The " + name + " KeySpace is ready");

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

public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

    } catch (KeyspacesException e) {
```

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

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateKeyspace](#)
 - [CreateTable](#)
 - [DeleteKeyspace](#)
 - [DeleteTable](#)
 - [GetKeyspace](#)
 - [GetTable](#)
 - [ListKeyspaces](#)
 - [ListTables](#)
 - [RestoreTable](#)
 - [UpdateTable](#)

Ejemplos de Kinesis que utilizan SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK for Java 2.x uso de Kinesis.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

- [Ejemplos sin servidor](#)

Acciones

CreateStream

En el siguiente ejemplo de código, se muestra cómo usar CreateStream.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateStream](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteStream

En el siguiente ejemplo de código, se muestra cómo usar DeleteStream.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteStream](#) la Referencia AWS SDK for Java 2.x de la API.

GetRecords

En el siguiente ejemplo de código, se muestra cómo usar GetRecords.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta [GetRecords](#) la Referencia AWS SDK for Java 2.x de la API.

PutRecord

En el siguiente ejemplo de código, se muestra cómo usar PutRecord.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.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 obtener más información sobre la API, consulta [PutRecord](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos sin servidor

Invocar una función de Lambda desde un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir registros de un flujo de Kinesis. La función recupera la carga útil de Kinesis, la decodifica desde Base64 y registra el contenido del registro.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de Kinesis con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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;
    }
}
```

```
}
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de un flujo de Kinesis. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Notificación de los errores de los elementos del lote de Kinesis con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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 ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with AWS KMS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Hola, clave KMS

El siguiente ejemplo de código muestra cómo empezar a utilizar la clave KMS.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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 HelloKMS {
    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();
        }
    }
}
```

```
        ListKeysIterable keysResponse =
kmsClient.listKeysPaginator(listKeysRequest);
        keysResponse.stream()
            .flatMap(r -> r.keys().stream())
            .forEach(key -> System.out
                .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
" + key.keyId()));
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Para obtener más información sobre la API, consulta [listKeysPaginator](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateAlias

En el siguiente ejemplo de código, se muestra cómo usar CreateAlias.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
    try {
```

```
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        kmsClient.createAlias(aliasRequest);
        System.out.println(aliasName + " was successfully created.");

    } catch (ResourceExistsException e) {
        System.err.println("Alias already exists: " + e.getMessage());
        System.err.println("Moving on...");
    } catch (Exception e) {
        System.err.println("An unexpected error occurred: " + e.getMessage());
        System.err.println("Moving on...");
    }
}
```

- Para obtener más información sobre la API, consulta [CreateAlias](#) la Referencia AWS SDK for Java 2.x de la API.

CreateGrant

En el siguiente ejemplo de código, se muestra cómo usar CreateGrant.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String grantKey(KmsClient kmsClient, String keyId, String
granteePrincipal) {
    try {
        // Add the desired KMS Grant permissions.
        List<GrantOperation> grantPermissions = new ArrayList<>();
        grantPermissions.add(GrantOperation.ENCRYPT);
        grantPermissions.add(GrantOperation.DECRYPT);
        grantPermissions.add(GrantOperation.DESCRIBE_KEY);
```

```
        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

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

- Para obtener más información sobre la API, consulta [CreateGrant](#) la Referencia AWS SDK for Java 2.x de la API.

CreateKey

En el siguiente ejemplo de código, se muestra cómo usar CreateKey.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createKey(KmsClient kmsClient, String keyDesc) {
    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();
```

```
        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.println("Symmetric key with ARN [" +
result.keyMetadata().arn() + "] has been created.");
        return result.keyMetadata().keyId();

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

- Para obtener más información sobre la API, consulta [CreateKey](#) la Referencia AWS SDK for Java 2.x de la API.

Decrypt

En el siguiente ejemplo de código, se muestra cómo usar Decrypt.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);

    } catch (KmsException e) {
```

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

- Para obtener información sobre la API, consulte [Decrypt](#) en la Referencia de la API de AWS SDK for Java 2.x .

DeleteAlias

En el siguiente ejemplo de código, se muestra cómo usar DeleteAlias.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
    try {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        kmsClient.deleteAlias(deleteAliasRequest);

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

- Para obtener más información sobre la API, consulta [DeleteAlias](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeKey

En el siguiente ejemplo de código, se muestra cómo usar DescribeKey.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        KeyState keyState = response.keyMetadata().keyState();
        if (keyState == KeyState.ENABLED) {
            System.out.println("The key is enabled.");
            return true;
        } else {
            System.out.println("The key is not enabled. Key state: " +
keyState);
        }

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

- Para obtener más información sobre la API, consulta [DescribeKey](#) la Referencia AWS SDK for Java 2.x de la API.

DisableKey

En el siguiente ejemplo de código, se muestra cómo usar `DisableKey`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void disableKey(KmsClient kmsClient, String keyId) {
    try {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.disableKey(keyRequest);

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

- Para obtener más información sobre la API, consulta [DisableKey](#) la Referencia AWS SDK for Java 2.x de la API.

EnableKey

En el siguiente ejemplo de código, se muestra cómo usar `EnableKey`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Enable the KMS key.
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 obtener más información sobre la API, consulta [EnableKey](#) la Referencia AWS SDK for Java 2.x de la API.

Encrypt

En el siguiente ejemplo de código, se muestra cómo usar Encrypt.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String
text) {
    try {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        EncryptResponse response = kmsClient.encrypt(encryptRequest);
    }
}
```

```
String algorithm = response.encryptionAlgorithm().toString();
System.out.println("The string was encrypted with algorithm " +
algorithm + ".");

// Get the encrypted data.
SdkBytes encryptedData = response.ciphertextBlob();
return encryptedData;

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

- Para obtener información sobre la API, consulte [Encrypt](#) en la referencia de la API de AWS SDK for Java 2.x .

ListAliases

En el siguiente ejemplo de código, se muestra cómo usar ListAliases.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void listAllAliases(KmsClient kmsClient) {
    try {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesIterable aliasesResponse =
kmsClient.listAliasesPaginator(aliasesRequest);
        aliasesResponse.stream()
            .flatMap(r -> r.aliases().stream())
```

```

        .forEach(alias -> System.out
            .println("The alias name is: " + alias.aliasName()));

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

```

- Para obtener más información sobre la API, consulta [ListAliases](#) la Referencia AWS SDK for Java 2.x de la API.

ListGrants

En el siguiente ejemplo de código, se muestra cómo usar ListGrants.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

public static void displayGrantIds(KmsClient kmsClient, String keyId) {
    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsIterable response =
kmsClient.listGrantsPaginator(grantsRequest);
        response.stream()
            .flatMap(r -> r.grants().stream())
            .forEach(grant -> {
                System.out.println("The grant Id is : " + grant.grantId());
                List<GrantOperation> ops = grant.operations();
                for (GrantOperation op : ops) {
                    System.out.println(op.name());
                }
            });
    }
}

```

```
        }
    });

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

- Para obtener más información sobre la API, consulta [ListGrants](#) la Referencia AWS SDK for Java 2.x de la API.

ListKeyPolicies

En el siguiente ejemplo de código, se muestra cómo usar `ListKeyPolicies`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
policyName) {
    try {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
        System.out.println("The response is "+response.policy());
    } catch (KmsException e) {
        if (e.getMessage().contains("No such policy exists")) {
            System.out.println("The policy cannot be found. Error message: " +
e.getMessage());
        } else {
            throw e;
        }
    }
}
```

```
    }  
  }  
}
```

- Para obtener más información sobre la API, consulta [ListKeyPolicies](#) la Referencia AWS SDK for Java 2.x de la API.

ListKeys

En el siguiente ejemplo de código, se muestra cómo usar `ListKeys`.

SDK para Java 2.x

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.kms.KmsClient;  
import software.amazon.awssdk.services.kms.model.ListKeysRequest;  
import software.amazon.awssdk.services.kms.model.KmsException;  
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including 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 HelloKMS {  
    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();

            ListKeysIterable keysResponse =
kmsClient.listKeysPaginator(listKeysRequest);
            keysResponse.stream()
                .flatMap(r -> r.keys().stream())
                .forEach(key -> System.out
                    .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
" + key.keyId()));

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

- Para obtener más información sobre la API, consulta [ListKeys](#) la Referencia AWS SDK for Java 2.x de la API.

RevokeGrant

En el siguiente ejemplo de código, se muestra cómo usar RevokeGrant.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
grantId) {
    try {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        kmsClient.revokeGrant(grantRequest);
        System.out.println("Grant ID: [" + grantId + "] was successfully
revoke!");

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

- Para obtener más información sobre la API, consulta [RevokeGrant](#) la Referencia AWS SDK for Java 2.x de la API.

ScheduleKeyDeletion

En el siguiente ejemplo de código, se muestra cómo usar ScheduleKeyDeletion.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteKey(KmsClient kmsClient, String keyId) {
    try {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();
```



```
kmsClient.scheduleKeyDeletion(deletionRequest);
System.out.println("The key will be deleted in 7 days.");

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

- Para obtener más información sobre la API, consulta [ScheduleKeyDeletion](#) la Referencia AWS SDK for Java 2.x de la API.

Sign

En el siguiente ejemplo de código, se muestra cómo usar Sign.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void signVerifyData(KmsClient kmsClient) {
    String signMessage = "Here is the message that will be digitally signed";

    // Create an AWS KMS key used to digitally sign data.
    CreateKeyRequest request = CreateKeyRequest.builder()
        .keySpec(KeySpec.RSA_2048) // Specify key spec
        .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
        .origin(OriginType.AWS_KMS) // Specify key origin
        .build();

    CreateKeyResponse response = kmsClient.createKey(request);
    String keyId2 = response.keyMetadata().keyId();
    System.out.println("Created KMS key with ID: " + keyId2);

    SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
}
```

```
SignRequest signRequest = SignRequest.builder()
    .keyId(keyId2)
    .message(bytes)
    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build();

SignResponse signResponse = kmsClient.sign(signRequest);
byte[] signedBytes = signResponse.signature().asByteArray();

// Verify the digital signature.
VerifyRequest verifyRequest = VerifyRequest.builder()
    .keyId(keyId2)

    .message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))
    .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build();

VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
}
```

- Para obtener información sobre la API, consulta la referencia de la AWS SDK for Java 2.x API de inicio de [sesión](#).

TagResource

En el siguiente ejemplo de código, se muestra cómo usar TagResource.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void tagKMSKey(KmsClient kmsClient, String keyId) {
    try {
```

```
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        kmsClient.tagResource(tagResourceRequest);
        System.out.println("The key has been tagged.");

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

- Para obtener más información sobre la API, consulta [TagResource](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Conozca las principales operaciones de KMS

En el siguiente ejemplo de código, se muestra cómo:

- Creación de una clave de KMS.
- Enumerar las claves de KMS de su cuenta y obtener información sobre ellas.
- Habilitar y desactivar claves KMS.
- Generar una clave de datos simétrica que pueda usarse para el cifrado del cliente.
- Genere una clave asimétrica que se utilice para firmar digitalmente los datos.
- Etiquete las claves.
- Eliminar las claves KMS.

SDK para Java 2.x

 Note

Hay más en marcha GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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.AliasListEntry;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.CreateAliasRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantResponse;
import software.amazon.awssdk.services.kms.model.CreateKeyRequest;
import software.amazon.awssdk.services.kms.model.CreateKeyResponse;
import software.amazon.awssdk.services.kms.model.CustomerMasterKeySpec;
import software.amazon.awssdk.services.kms.model.DecryptRequest;
import software.amazon.awssdk.services.kms.model.DecryptResponse;
import software.amazon.awssdk.services.kms.model.DeleteAliasRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyResponse;
import software.amazon.awssdk.services.kms.model.DisableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationRequest;
import software.amazon.awssdk.services.kms.model.EncryptRequest;
import software.amazon.awssdk.services.kms.model.EncryptResponse;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyResponse;
import software.amazon.awssdk.services.kms.model.GrantOperation;
import software.amazon.awssdk.services.kms.model.KeySpec;
import software.amazon.awssdk.services.kms.model.KeyState;
import software.amazon.awssdk.services.kms.model.KeyUsageType;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.LimitExceededException;
import software.amazon.awssdk.services.kms.model.ListAliasesRequest;
import software.amazon.awssdk.services.kms.model.ListGrantsRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesResponse;
import software.amazon.awssdk.services.kms.model.OriginType;
```

```
import software.amazon.awssdk.services.kms.model.PutKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.RevokeGrantRequest;
import software.amazon.awssdk.services.kms.model.ScheduleKeyDeletionRequest;
import software.amazon.awssdk.services.kms.model.SignRequest;
import software.amazon.awssdk.services.kms.model.SignResponse;
import software.amazon.awssdk.services.kms.model.SigningAlgorithmSpec;
import software.amazon.awssdk.services.kms.model.Tag;
import software.amazon.awssdk.services.kms.model.TagResourceRequest;
import software.amazon.awssdk.services.kms.model.VerifyRequest;
import software.amazon.awssdk.services.kms.model.VerifyResponse;
import software.amazon.awssdk.services.kms.paginators.ListAliasesIterable;
import software.amazon.awssdk.services.kms.paginators.ListGrantsIterable;
import software.amazon.awssdk.services.secretsmanager.model.ResourceExistsException;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;
import java.nio.ByteBuffer;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.ArrayList;
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
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        final String usage = ""
            Usage: <granteePrincipal>

            Where:
                granteePrincipal - The principal (user, service account, or
                group) to whom the grant or permission is being given.
        """;
    }
}
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }
    String granteePrincipal = args[0];
    String policyName = "default";

    Scanner scanner = new Scanner(System.in);
    String keyDesc = "Created by the AWS KMS API";

    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("""
        Welcome to the AWS Key Management SDK Getting Started scenario.

        This program demonstrates how to interact with AWS Key Management using
the AWS SDK for Java (v2).
        The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
        and manage AWS KMS keys and control their use across a wide range of AWS
services and applications.
        KMS provides a centralized and unified approach to managing encryption
keys, making it easier to meet your
        data protection and regulatory compliance requirements.

        This Getting Started scenario creates two key types. A symmetric
encryption key is used to encrypt and decrypt data,
        and an asymmetric key used to digitally sign data.
        Let's get started...
        """);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("1. Create a symmetric KMS key\n");
    System.out.println("First, the program will creates a symmetric KMS key that
you can used to encrypt and decrypt data.");
    waitForInputToContinue(scanner);
    String targetKeyId = createKey(kmsClient, keyDesc);
    waitForInputToContinue(scanner);
```

```

System.out.println(DASHES);
System.out.println("""
    2. Enable a KMS key

```

By default, when the SDK creates an AWS key it is enabled. The next bit of code checks to determine if the key is enabled. If it is not enabled, the code enables it.

```

    """);
waitForInputToContinue(scanner);
boolean isEnabled = isKeyEnabled(kmsClient, targetKeyId);
if (!isEnabled)
    enableKey(kmsClient, targetKeyId);
waitForInputToContinue(scanner);

```

```

System.out.println(DASHES);
System.out.println("3. Encrypt data using the symmetric KMS key");
String plaintext = "Hello, AWS KMS!";
System.out.printf("""

```

One of the main uses of symmetric keys is to encrypt and decrypt data.

Next, the code encrypts the string '%s' with the SYMMETRIC_DEFAULT encryption algorithm.

```

    %n""", plaintext);
waitForInputToContinue(scanner);
SdkBytes ciphertext = encryptData(kmsClient, targetKeyId, plaintext);
waitForInputToContinue(scanner);

```

```

System.out.println(DASHES);
System.out.println("4. Create an alias");
System.out.println("""

```

Enter an alias name for the key. The name should be prefixed with 'alias/'.

For example, 'alias/myFirstKey'.

```

    """);

```

```

String aliasName = scanner.nextLine();
String fullAliasName = aliasName.isEmpty() ? "alias/dev-encryption-key" :
aliasName;
createCustomAlias(kmsClient, targetKeyId, fullAliasName);
waitForInputToContinue(scanner);

```

```

System.out.println(DASHES);
System.out.println("5. List all of your aliases");
waitForInputToContinue(scanner);
listAllAliases(kmsClient);
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("6. Enable automatic rotation of the KMS key");
System.out.println("

```

By default, when the SDK enables automatic rotation of a KMS key, KMS rotates the key material of the KMS key one year (approximately 365 days) from the enable date and every year thereafter.

```

""");
waitForInputToContinue(scanner);
enableKeyRotation(kmsClient, targetKeyId);
waitForInputToContinue(scanner);

```

```

System.out.println(DASHES);
System.out.println("
    7. Create a grant

```

A grant is a policy instrument that allows Amazon Web Services principals to use KMS keys.

It also can allow them to view a KMS key (DescribeKey) and create and manage grants.

When authorizing access to a KMS key, grants are considered along with key policies and IAM policies.

```

""");

waitForInputToContinue(scanner);
String grantId = grantKey(kmsClient, targetKeyId, granteePrincipal);
System.out.println("The code granted principal with ARN [" +
granteePrincipal + "] ");
System.out.println("use of the symmetric key. The grant ID is [" + grantId +
"]");
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("8. List grants for the KMS key");
waitForInputToContinue(scanner);
displayGrantIds(kmsClient, targetKeyId);
waitForInputToContinue(scanner);

```



```
System.out.println(DASHES);
System.out.println("9. Revoke the grant");
waitForInputToContinue(scanner);
revokeKeyGrant(kmsClient, targetKeyId, grantId);
waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
System.out.println("10. Decrypt the data\n");
System.out.println("""
```

Lets decrypt the data that was encrypted in an early step.
The code uses the same key to decrypt the string that we encrypted earlier in the program.

```
""");
waitForInputToContinue(scanner);
String decryptText = decryptData(kmsClient, ciphertext, targetKeyId);
System.out.println("Decrypted text is: " + decryptText);
waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
System.out.println("11. Replace a key policy\n");
System.out.println("""
```

A key policy is a resource policy for a KMS key. Key policies are the primary way to control access to KMS keys. Every KMS key must have exactly one key policy. The statements in the key policy determine who has permission to use the KMS key and how they can use it.

You can also use IAM policies and grants to control access to the KMS key, but every KMS key must have a key policy.

By default, when you create a key by using the SDK, a policy is created that gives the AWS account that owns the KMS key full access to the KMS key.

Let's try to replace the automatically created policy with the following policy.

```
"Version": "2012-10-17",
"Statement": [{
"Effect": "Allow",
"Principal": {"AWS": "arn:aws:iam::0000000000:root"},
"Action": "kms:*",
```

```
        "Resource": "*"
    }]
    """);

    waitForInputToContinue(scanner);
    boolean polAdded = replacePolicy(kmsClient, targetKeyId, policyName);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("12. Get the key policy\n");
    System.out.println("The next bit of code that runs gets the key policy to
make sure it exists.");
    waitForInputToContinue(scanner);
    getKeyPolicy(kmsClient, targetKeyId, policyName);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("13. Create an asymmetric KMS key and sign your data\n");
    System.out.println("""
        Signing your data with an AWS key can provide several benefits that make
it an attractive option
        for your data signing needs. By using an AWS KMS key, you can leverage
the
        security controls and compliance features provided by AWS,
        which can help you meet various regulatory requirements and enhance the
overall security posture
        of your organization.
    """);
    waitForInputToContinue(scanner);
    signVerifyData(kmsClient);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("14. Tag your symmetric KMS Key\n");
    System.out.println("""
        By using tags, you can improve the overall management, security, and
governance of your
        KMS keys, making it easier to organize, track, and control access to
your encrypted data within
        your AWS environment
    """);
    waitForInputToContinue(scanner);
    tagKMSKey(kmsClient, targetKeyId);
    waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
System.out.println("15. Schedule the deletion of the KMS key\n");
System.out.println("""
    By default, KMS applies a waiting period of 30 days,
    but you can specify a waiting period of 7-30 days. When this operation
is successful,
    the key state of the KMS key changes to PendingDeletion and the key
can't be used in any
    cryptographic operations. It remains in this state for the duration of
the waiting period.

    Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
    all data that was encrypted under the KMS key is unrecoverable.\s
""");
System.out.println("Would you like to delete the Key Management resources?
(y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    System.out.println("You selected to delete the AWS KMS resources.");
    waitForInputToContinue(scanner);
    deleteSpecificAlias(kmsClient, fullAliasName);
    disableKey(kmsClient, targetKeyId);
    deleteKey(kmsClient, targetKeyId);

} else {
    System.out.println("The Key Management resources will not be deleted");
}

System.out.println(DASHES);
System.out.println("This concludes the AWS Key Management SDK Getting
Started scenario");
System.out.println(DASHES);
}
public static void listAllAliases(KmsClient kmsClient) {
    try {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesIterable aliasesResponse =
kmsClient.listAliasesPaginator(aliasesRequest);
        aliasesResponse.stream()
```

```
        .flatMap(r -> r.aliasNames().stream())
        .forEach(alias -> System.out
            .println("The alias name is: " + alias.aliasName()));

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

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);
    }
}

public static void signVerifyData(KmsClient kmsClient) {
    String signMessage = "Here is the message that will be digitally signed";

    // Create an AWS KMS key used to digitally sign data.
    CreateKeyRequest request = CreateKeyRequest.builder()
        .keySpec(KeySpec.RSA_2048) // Specify key spec
        .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
        .origin(OriginType.AWS_KMS) // Specify key origin
        .build();

    CreateKeyResponse response = kmsClient.createKey(request);
    String keyId2 = response.keyMetadata().keyId();
    System.out.println("Created KMS key with ID: " + keyId2);

    SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
    SignRequest signRequest = SignRequest.builder()
        .keyId(keyId2)
        .message(bytes)
        .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
        .build();
}
```

```
SignResponse signResponse = kmsClient.sign(signRequest);
byte[] signedBytes = signResponse.signature().asByteArray();

// Verify the digital signature.
VerifyRequest verifyRequest = VerifyRequest.builder()
    .keyId(keyId2)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset()))
    .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build());

VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
}

public static void tagKMSKey(KmsClient kmsClient, String keyId) {
    try {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        kmsClient.tagResource(tagResourceRequest);
        System.out.println("The key has been tagged.");

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

public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
policyName) {
    try {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
```

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

        GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
        System.out.println("The response is "+response.policy());
    } catch (KmsException e) {
        if (e.getMessage().contains("No such policy exists")) {
            System.out.println("The policy cannot be found. Error message: " +
e.getMessage());
        } else {
            throw e;
        }
    }
}

public static boolean replacePolicy(KmsClient kmsClient, String keyId, String
policyName) {
    // Change the principle in the below JSON.
    String policy = """"
    {
        "Version": "2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::%s:root"},
            "Action": "kms:*",
            "Resource": "*"
        }]
    }
    """".formatted(accountId);

    try {
        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .policy(policy)
            .build();
        kmsClient.putKeyPolicy(keyPolicyRequest);
        System.out.println("The key policy has been replaced.");
    } catch (LimitExceededException e) {
        System.out.println("Policy limit reached. Unable to create the
policy.");
        return false;
    } catch (AlreadyExistsException e) {
```

```
        System.out.println("Only one policy per key is supported. Unable to
create the policy.");
        return false;
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return true;
}

public static boolean doesKeyHavePolicy(KmsClient kmsClient, String keyId,
String policyName){
    ListKeyPoliciesRequest policiesRequest = ListKeyPoliciesRequest.builder()
        .keyId(keyId)
        .build();

    boolean hasPolicy = false;
    ListKeyPoliciesResponse response =
kmsClient.listKeyPolicies(policiesRequest);
    List<String>policyNames = response.policyNames();
    for (String pol : policyNames) {
        hasPolicy = true;
    }
    return hasPolicy;
}

public static void deleteKey(KmsClient kmsClient, String keyId) {
    try {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();

        kmsClient.scheduleKeyDeletion(deletionRequest);
        System.out.println("The key will be deleted in 7 days.");

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

```
public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
    try {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        kmsClient.deleteAlias(deleteAliasRequest);

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

public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        KeyState keyState = response.keyMetadata().keyState();
        if (keyState == KeyState.ENABLED) {
            System.out.println("The key is enabled.");
            return true;
        } else {
            System.out.println("The key is not enabled. Key state: " +
keyState);
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return false;
}

public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();
```



```
        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);

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

public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
grantId) {
    try {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        kmsClient.revokeGrant(grantRequest);
        System.out.println("Grant ID: [" + grantId + "] was successfully
revoke!");

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

public static void displayGrantIds(KmsClient kmsClient, String keyId) {
    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsIterable response =
kmsClient.listGrantsPaginator(grantsRequest);
        response.stream()
            .flatMap(r -> r.grants().stream())
            .forEach(grant -> {
                System.out.println("The grant Id is : " + grant.grantId());
                List<GrantOperation> ops = grant.operations();
                for (GrantOperation op : ops) {
```

```
        System.out.println(op.name());
    }
});

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

}

public static String grantKey(KmsClient kmsClient, String keyId, String
granteePrincipal) {
    try {
        // Add the desired KMS Grant permissions.
        List<GrantOperation> grantPermissions = new ArrayList<>();
        grantPermissions.add(GrantOperation.ENCRYPT);
        grantPermissions.add(GrantOperation.DECRYPT);
        grantPermissions.add(GrantOperation.DESCRIBE_KEY);

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

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

public static void enableKeyRotation(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRotationRequest enableKeyRotationRequest =
        EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKeyRotation(enableKeyRotationRequest);
    }
}
```

```
        System.out.println("Key rotation has been enabled for key with id [" +
keyId + "]);

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

    public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
        try {
            CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
                .aliasName(aliasName)
                .targetKeyId(targetKeyId)
                .build();

            kmsClient.createAlias(aliasRequest);
            System.out.println(aliasName + " was successfully created.");

        } catch (ResourceExistsException e) {
            System.err.println("Alias already exists: " + e.getMessage());
            System.err.println("Moving on...");
        } catch (Exception e) {
            System.err.println("An unexpected error occurred: " + e.getMessage());
            System.err.println("Moving on...");
        }
    }

    public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String
text) {
        try {
            SdkBytes myBytes = SdkBytes.fromUtf8String(text);
            EncryptRequest encryptRequest = EncryptRequest.builder()
                .keyId(keyId)
                .plaintext(myBytes)
                .build();

            EncryptResponse response = kmsClient.encrypt(encryptRequest);
            String algorithm = response.encryptionAlgorithm().toString();
            System.out.println("The string was encrypted with algorithm " +
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 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.println("Symmetric key with ARN [" +
result.keyMetadata().arn() + "] has been created.");
        return result.keyMetadata().keyId();

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

// Enable the KMS key.
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);
    }
}
```

```
private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        System.out.println("");
        System.out.println("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            System.out.println("Continuing with the program...");
            System.out.println("");
            break;
        } else {
            // Handle invalid input.
            System.out.println("Invalid input. Please try again.");
        }
    }
}

private static String getAccountId(){
    try (StsClient stsClient = StsClient.create()){
        GetCallerIdentityResponse callerIdentity =
stsClient.getCallerIdentity();
        return callerIdentity.account();
    }
}
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateKey](#)
 - [DescribeKey](#)
 - [DisableKey](#)
 - [EnableKey](#)
 - [GenerateDataKey](#)
 - [ListKeys](#)
 - [ScheduleKeyDeletion](#)
 - [Sign](#)

Ejemplos de Lambda usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Lambda.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Lambda

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Lambda.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListFunctions](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)
- [Ejemplos sin servidor](#)

Acciones

CreateFunction

En el siguiente ejemplo de código, se muestra cómo usar CreateFunction.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateFunction](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteFunction

En el siguiente ejemplo de código, se muestra cómo usar DeleteFunction.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteFunction](#) la Referencia AWS SDK for Java 2.x de la API.

Invoke

En el siguiente ejemplo de código, se muestra cómo usar Invoke.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener información sobre la API, consulte [Invocar](#) en la referencia de la API de AWS SDK for Java 2.x .

Escenarios

Comenzar a usar las funciones

En el siguiente ejemplo de código, se muestra cómo:

- Crear un rol de IAM y una función de Lambda y, a continuación, cargar el código de controlador.
- Invocar la función con un único parámetro y obtener resultados.
- Actualizar el código de la función y configurar con una variable de entorno.

- Invocar la función con un nuevo parámetro y obtener resultados. Mostrar el registro de ejecución devuelto.
- Enumerar las funciones de su cuenta y, luego, limpiar los recursos.

Para obtener información, consulte [Crear una función de Lambda con la consola](#).

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)

- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

Ejemplos sin servidor

Conexión a una base de datos de Amazon RDS en una función de Lambda

En el siguiente ejemplo de código, se muestra cómo implementar una función de Lambda que se conecta a una base de datos de RDS. La función realiza una solicitud sencilla a la base de datos y devuelve el resultado.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Conexión a una base de datos de Amazon RDS en una función Lambda mediante Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements RequestHandler<APIGatewayProxyRequestEvent,
    APIGatewayProxyResponseEvent> {
```

```
@Override
public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
event, Context context) {
    APIGatewayProxyResponseEvent response = new APIGatewayProxyResponseEvent();

    try {
        // Obtain auth token
        String token = createAuthToken();

        // Define connection configuration
        String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
            System.getenv("ProxyHostName"),
            System.getenv("Port"),
            System.getenv("DBName"));

        // Establish a connection to the database
        try (Connection connection =
DriverManager.getConnection(connectionString, System.getenv("DBUserName"), token);
            PreparedStatement statement = connection.prepareStatement("SELECT ?
+ ? AS sum")) {

            statement.setInt(1, 3);
            statement.setInt(2, 2);

            try (ResultSet resultSet = statement.executeQuery()) {
                if (resultSet.next()) {
                    int sum = resultSet.getInt("sum");
                    response.setStatusCode(200);
                    response.setBody("The selected sum is: " + sum);
                }
            }
        }

    } catch (Exception e) {
        response.setStatusCode(500);
        response.setBody("Error: " + e.getMessage());
    }

    return response;
}

private String createAuthToken() {
```

```

// Create RDS Data Service client
RdsDataClient rdsDataClient = RdsDataClient.builder()
    .region(Region.of(System.getenv("AWS_REGION")))
    .credentialsProvider(DefaultCredentialsProvider.create())
    .build();

// Define authentication request
ExecuteStatementRequest request = ExecuteStatementRequest.builder()
    .resourceArn(System.getenv("ProxyHostName"))
    .secretArn(System.getenv("DBUserName"))
    .database(System.getenv("DBName"))
    .sql("SELECT 'RDS IAM Authentication'")
    .build();

// Execute request and obtain authentication token
ExecuteStatementResponse response = rdsDataClient.executeStatement(request);
Field tokenField = response.records().get(0).get(0);

return tokenField.stringValue();
}
}

```

Invocar una función de Lambda desde un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir registros de un flujo de Kinesis. La función recupera la carga útil de Kinesis, la decodifica desde Base64 y registra el contenido del registro.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de Kinesis con Lambda mediante Java.

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

```



```
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;
    }
}
```

Invocación de una función de Lambda desde un desencadenador de DynamoDB

El siguiente ejemplo de código muestra cómo implementar una función de Lambda que recibe un evento desencadenado al recibir registros de una transmisión de DynamoDB. La función recupera la carga útil de DynamoDB y registra el contenido del registro.

SDK para Java 2.x

 Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Consumo de un evento de DynamoDB con Lambda mediante Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import
    com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " + GSON.toJson(record.getDynamodb()));
    }
}
```

Invocación de una función de Lambda desde un desencadenador de Amazon S3

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al cargar un objeto en un bucket de S3. La función recupera el nombre del bucket

de S3 y la clave del objeto del parámetro de evento y llama a la API de Amazon S3 para recuperar y registrar el tipo de contenido del objeto.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de S3 con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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 una función de Lambda desde un desencadenador de Amazon SNS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de un tema de SNS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de SNS con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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 una función de Lambda desde un desencadenador de Amazon SQS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de una cola de SQS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de SQS con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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;
        }
    }
}
```

```
    }  
  }  
}
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de un flujo de Kinesis. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Notificación de los errores de los elementos del lote de Kinesis con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
// SPDX-License-Identifier: Apache-2.0  
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);
    }
}

```

Notificación de los errores de los elementos del lote de las funciones de Lambda con un desencadenador de DynamoDB

El siguiente ejemplo de código muestra cómo implementar una respuesta por lotes parcial para las funciones de Lambda que reciben eventos de una transmisión de DynamoDB. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Notificación de los errores de los elementos del lote de DynamoDB con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
    Serializable> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
    {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
        ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
        input.getRecords()) {
            try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.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();
    }
}
```

```
}  
}
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Amazon SQS.

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de una cola de SQS. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Java 2.x

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Notificación de los errores de los elementos del lote de SQS con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
// SPDX-License-Identifier: Apache-2.0  
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 ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with MediaConvert.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

CreateJob

En el siguiente ejemplo de código, se muestra cómo usar CreateJob.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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()

                .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 obtener más información sobre la API, consulta [CreateJob](#) la Referencia AWS SDK for Java 2.x de la API.

GetJob

En el siguiente ejemplo de código, se muestra cómo usar GetJob.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetJob](#) la Referencia AWS SDK for Java 2.x de la API.

ListJobs

En el siguiente ejemplo de código, se muestra cómo usar ListJobs.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListJobs](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Migration Hub usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK for Java 2.x uso de Migration Hub.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

DeleteProgressUpdateStream

En el siguiente ejemplo de código, se muestra cómo usar DeleteProgressUpdateStream.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteProgressUpdateStream](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeApplicationState

En el siguiente ejemplo de código, se muestra cómo usar DescribeApplicationState.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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 obtener más información sobre la API, consulta [DescribeApplicationState](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeMigrationTask

En el siguiente ejemplo de código, se muestra cómo usar `DescribeMigrationTask`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeMigrationTask](#) la Referencia AWS SDK for Java 2.x de la API.

ImportMigrationTask

En el siguiente ejemplo de código, se muestra cómo usar `ImportMigrationTask`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ImportMigrationTask](#) la Referencia AWS SDK for Java 2.x de la API.

ListApplications

En el siguiente ejemplo de código, se muestra cómo usar `ListApplications`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListApplications](#) la Referencia AWS SDK for Java 2.x de la API.

ListCreatedArtifacts

En el siguiente ejemplo de código, se muestra cómo usar `ListCreatedArtifacts`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListCreatedArtifacts](#) la Referencia AWS SDK for Java 2.x de la API.

ListMigrationTasks

En el siguiente ejemplo de código, se muestra cómo usar ListMigrationTasks.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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 obtener más información sobre la API, consulta [ListMigrationTasks](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Amazon Personalize usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Amazon Personalize.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

CreateBatchInferenceJob

En el siguiente ejemplo de código, se muestra cómo usar CreateBatchInferenceJob.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                String solutionVersionArn,
                String jobName,
                String s3InputDataSourcePath,
                String s3DataDestinationPath,
                String roleArn,
                String explorationWeight,
                String explorationItemAgeCutOff) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String batchInferenceJobArn;

    try {

        // Set up data input and output parameters.
        S3DataConfig inputSource = S3DataConfig.builder()
            .path(s3InputDataSourcePath)
            .build();

        S3DataConfig outputDestination = S3DataConfig.builder()
            .path(s3DataDestinationPath)
            .build();

        BatchInferenceJobInput jobInput =
BatchInferenceJobInput.builder()
            .s3DataSource(inputSource)
            .build();

        BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
            .s3DataDestination(outputDestination)
            .build();

        // Optional code to build the User-Personalization specific
item exploration

        // config.
        HashMap<String, String> explorationConfig = new HashMap<>();

        explorationConfig.put("explorationWeight",
explorationWeight);
```



```
        explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

        BatchInferenceJobConfig jobConfig =
BatchInferenceJobConfig.builder()
                        .itemExplorationConfig(explorationConfig)
                        .build();

        // End optional User-Personalization recipe specific code.

        CreateBatchInferenceJobRequest
createBatchInferenceJobRequest = CreateBatchInferenceJobRequest
                        .builder()
                        .solutionVersionArn(solutionVersionArn)
                        .jobInput(jobInput)
                        .jobOutput(jobOutputLocation)
                        .jobName(jobName)
                        .roleArn(roleArn)
                        .batchInferenceJobConfig(jobConfig) //
Optional
                        .build();

        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
                        .batchInferenceJobArn();

        DescribeBatchInferenceJobRequest
describeBatchInferenceJobRequest = DescribeBatchInferenceJobRequest
                        .builder()
                        .batchInferenceJobArn(batchInferenceJobArn)
                        .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
        while (Instant.now().getEpochSecond() < maxTime) {

                BatchInferenceJob batchInferenceJob =
personalizeClient
                        .describeBatchInferenceJob(describeBatchInferenceJobRequest)
                                .batchInferenceJob();

                status = batchInferenceJob.status();
                System.out.println("Batch inference job status: " +
status);
```

```
        if (status.equals("ACTIVE") || status.equals("CREATE
FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return batchInferenceJobArn;

} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
```

- Para obtener más información sobre la API, consulta [CreateBatchInferenceJob](#) la Referencia AWS SDK for Java 2.x de la API.

CreateCampaign

En el siguiente ejemplo de código, se muestra cómo usar CreateCampaign.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateCampaign](#) la Referencia AWS SDK for Java 2.x de la API.

CreateDataset

En el siguiente ejemplo de código, se muestra cómo usar CreateDataset.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createDataset(PersonalizeClient personalizeClient,
    String datasetName,
    String datasetGroupArn,
    String datasetType,
    String schemaArn) {
    try {
        CreateDatasetRequest request = CreateDatasetRequest.builder()
```

```
        .name(datasetName)
        .datasetGroupArn(datasetGroupArn)
        .datasetType(datasetType)
        .schemaArn(schemaArn)
        .build();

    String datasetArn = personalizeClient.createDataset(request)
        .datasetArn();
    System.out.println("Dataset " + datasetName + " created.");
    return datasetArn;

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

- Para obtener más información sobre la API, consulta [CreateDataset](#) la Referencia AWS SDK for Java 2.x de la API.

CreateDatasetExportJob

En el siguiente ejemplo de código, se muestra cómo usar `CreateDatasetExportJob`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createDatasetExportJob(PersonalizeClient personalizeClient,
    String jobName,
    String datasetArn,
    IngestionMode ingestionMode,
    String roleArn,
    String s3BucketPath,
    String kmsKeyArn) {
```

```
long waitInMilliseconds = 30 * 1000; // 30 seconds
String status = null;

try {

    S3DataConfig exportS3DataConfig =
S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
    DatasetExportJobOutput jobOutput =
DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig)
        .build();

    CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
        .jobName(jobName)
        .datasetArn(datasetArn)
        .ingestionMode(ingestionMode)
        .jobOutput(jobOutput)
        .roleArn(roleArn)
        .build();

    String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

    DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
        .datasetExportJobArn(datasetExportJobArn)
        .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetExportJob datasetExportJob = personalizeClient
            .describeDatasetExportJob(describeDatasetExportJobRequest)
            .datasetExportJob();

        status = datasetExportJob.status();
        System.out.println("Export job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            return status;
        }
    }
}
```

```
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- Para obtener más información sobre la API, consulta [CreateDatasetExportJob](#) la Referencia AWS SDK for Java 2.x de la API.

CreateDatasetGroup

En el siguiente ejemplo de código, se muestra cómo usar CreateDatasetGroup.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .build();

        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

```
}
```

Crear un grupo de conjuntos de datos de dominios.

```
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 obtener más información sobre la API, consulta [CreateDatasetGroup](#) la Referencia AWS SDK for Java 2.x de la API.

CreateDatasetImportJob

En el siguiente ejemplo de código, se muestra cómo usar CreateDatasetImportJob.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
    String jobName,
    String datasetArn,
    String s3BucketPath,
    String roleArn) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String datasetImportJobArn;

    try {
        DataSource importDataSource = DataSource.builder()
            .dataLocation(s3BucketPath)
            .build();

        CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
            .datasetArn(datasetArn)
            .dataSource(importDataSource)
            .jobName(jobName)
            .roleArn(roleArn)
            .build();

        datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
            .datasetImportJobArn();

        DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
            .datasetImportJobArn(datasetImportJobArn)
            .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {

            DatasetImportJob datasetImportJob = personalizeClient
                .describeDatasetImportJob(describeDatasetImportJobRequest)
                .datasetImportJob();

            status = datasetImportJob.status();
            System.out.println("Dataset import job status: " + status);
        }
    }
}
```



```

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return datasetImportJobArn;

} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}

```

- Para obtener más información sobre la API, consulta [CreateDatasetImportJob](#) la Referencia AWS SDK for Java 2.x de la API.

CreateEventTracker

En el siguiente ejemplo de código, se muestra cómo usar CreateEventTracker.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

public static String createEventTracker(PersonalizeClient personalizeClient,
String eventTrackerName,
    String datasetGroupArn) {

    String eventTrackerId = "";
    String eventTrackerArn;
    long maxTime = 3 * 60 * 60; // 3 hours
    long waitInMilliseconds = 20 * 1000; // 20 seconds

```

```
String status;

try {

    CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
        .name(eventTrackerName)
        .datasetGroupArn(datasetGroupArn)
        .build();

    CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient
        .createEventTracker(createEventTrackerRequest);

    eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
    eventTrackerId = createEventTrackerResponse.trackingId();
    System.out.println("Event tracker ARN: " + eventTrackerArn);
    System.out.println("Event tracker ID: " + eventTrackerId);

    maxTime = Instant.now().getEpochSecond() + maxTime;

    DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
        .eventTrackerArn(eventTrackerArn)
        .build();

    while (Instant.now().getEpochSecond() < maxTime) {

        status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
        System.out.println("EventTracker status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return eventTrackerId;
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
```

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

- Para obtener más información sobre la API, consulta [CreateEventTracker](#) la Referencia AWS SDK for Java 2.x de la API.

CreateFilter

En el siguiente ejemplo de código, se muestra cómo usar CreateFilter.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createFilter(PersonalizeClient personalizeClient,
    String filterName,
    String datasetGroupArn,
    String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
            .name(filterName)
            .datasetGroupArn(datasetGroupArn)
            .filterExpression(filterExpression)
            .build();

        return personalizeClient.createFilter(request).filterArn();
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Para obtener más información sobre la API, consulta [CreateFilter](#) la Referencia AWS SDK for Java 2.x de la API.

CreateRecommender

En el siguiente ejemplo de código, se muestra cómo usar CreateRecommender.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createRecommender(PersonalizeClient personalizeClient,
    String name,
    String datasetGroupArn,
    String recipeArn) {

    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String recommenderStatus = "";

    try {
        CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
            .datasetGroupArn(datasetGroupArn)
            .name(name)
            .recipeArn(recipeArn)
            .build();

        CreateRecommenderResponse recommenderResponse = personalizeClient
            .createRecommender(createRecommenderRequest);
        String recommenderArn = recommenderResponse.recommenderArn();
        System.out.println("The recommender ARN is " + recommenderArn);

        DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
            .recommenderArn(recommenderArn)
            .build();
```

```
maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

while (Instant.now().getEpochSecond() < maxTime) {

    recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
        .status();
    System.out.println("Recommender status: " + recommenderStatus);

    if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
        break;
    }
    try {
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
return recommenderArn;

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

- Para obtener más información sobre la API, consulta [CreateRecommender](#) la Referencia AWS SDK for Java 2.x de la API.

CreateSchema

En el siguiente ejemplo de código, se muestra cómo usar CreateSchema.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

    String schema = null;
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .schema(schema)
            .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;

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

Crea un esquema con un dominio.

```
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 obtener más información sobre la API, consulte [CreateSchema](#) la referencia AWS SDK for Java 2.x de la API.

CreateSolution

En el siguiente ejemplo de código, se muestra cómo usar CreateSolution.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
        String datasetGroupArn,
        String solutionName,
        String recipeArn) {

    try {
        CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
            .name(solutionName)
            .datasetGroupArn(datasetGroupArn)
            .recipeArn(recipeArn)
            .build();

        CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
        return solutionResponse.solutionArn();

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

- Para obtener más información sobre la API, consulta [CreateSolution](#) la Referencia AWS SDK for Java 2.x de la API.

CreateSolutionVersion

En el siguiente ejemplo de código, se muestra cómo usar `CreateSolutionVersion`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String solutionStatus = "";
    String solutionVersionStatus = "";
    String solutionVersionArn = "";

    try {
        DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        // Wait until solution is active.
        while (Instant.now().getEpochSecond() < maxTime) {

            solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
            System.out.println("Solution status: " + solutionStatus);

            if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

```
        if (solutionStatus.equals("ACTIVE")) {

            CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
                .solutionArn(solutionArn)
                .build();

            CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
                .createSolutionVersion(createSolutionVersionRequest);
            solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

            System.out.println("Solution version ARN: " + solutionVersionArn);

            DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
                .solutionVersionArn(solutionVersionArn)
                .build();

            while (Instant.now().getEpochSecond() < maxTime) {

                solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                    .solutionVersion().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 obtener más información sobre la API, consulta [CreateSolutionVersion](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteCampaign

En el siguiente ejemplo de código, se muestra cómo usar DeleteCampaign.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,  
String campaignArn) {  
  
    try {  
        DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()  
            .campaignArn(campaignArn)  
            .build();  
  
        personalizeClient.deleteCampaign(campaignRequest);  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obtener más información sobre la API, consulta [DeleteCampaign](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteEventTracker

En el siguiente ejemplo de código, se muestra cómo usar DeleteEventTracker.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
    try {
        DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode();

        System.out.println("Status code:" + status);


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

- Para obtener más información sobre la API, consulta [DeleteEventTracker](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteSolution

En el siguiente ejemplo de código, se muestra cómo usar DeleteSolution.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteGivenSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        personalizeClient.deleteSolution(solutionRequest);
        System.out.println("Done");


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

- Para obtener más información sobre la API, consulta [DeleteSolution](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeCampaign

En el siguiente ejemplo de código, se muestra cómo usar DescribeCampaign.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

    try {
        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign myCampaign = campaignResponse.campaign();
        System.out.println("The Campaign name is " + myCampaign.name());
        System.out.println("The Campaign status is " + myCampaign.status());

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

- Para obtener más información sobre la API, consulta [DescribeCampaign](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeRecipe

En el siguiente ejemplo de código, se muestra cómo usar DescribeRecipe.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {

    try {
```

```
DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
    .recipeArn(recipeArn)
    .build();

DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
System.out.println("The recipe name is " +
recipeResponse.recipe().name());

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

- Para obtener más información sobre la API, consulta [DescribeRecipe](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeSolution

En el siguiente ejemplo de código, se muestra cómo usar DescribeSolution.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeSpecificSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DescribeSolutionRequest solutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();
```

```
        DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
        System.out.println("The Solution name is " +
response.solution().name());

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

- Para obtener más información sobre la API, consulta [DescribeSolution](#) la Referencia AWS SDK for Java 2.x de la API.

ListCampaigns

En el siguiente ejemplo de código, se muestra cómo usar `ListCampaigns`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void listAllCampaigns(PersonalizeClient personalizeClient, String
solutionArn) {

    try {
        ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
            .maxResults(10)
            .solutionArn(solutionArn)
            .build();

        ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
        List<CampaignSummary> campaigns = response.campaigns();
        for (CampaignSummary campaign : campaigns) {
            System.out.println("Campaign name is : " + campaign.name());
        }
    }
}
```



```

        System.out.println("Campaign ARN is : " + campaign.campaignArn());
    }

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

```

- Para obtener más información sobre la API, consulta [ListCampaigns](#) la Referencia AWS SDK for Java 2.x de la API.

ListDatasetGroups

En el siguiente ejemplo de código, se muestra cómo usar `ListDatasetGroups`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

public static void listDSGroups(PersonalizeClient personalizeClient) {

    try {
        ListDatasetGroupsRequest groupsRequest =
ListDatasetGroupsRequest.builder()
        .maxResults(15)
        .build();

        ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
        List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
        for (DatasetGroupSummary group : groups) {
            System.out.println("The DataSet name is : " + group.name());
            System.out.println("The DataSet ARN is : " +
group.datasetGroupArn());
        }
    }
}

```

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

- Para obtener más información sobre la API, consulta [ListDatasetGroups](#) la Referencia AWS SDK for Java 2.x de la API.

ListRecipes

En el siguiente ejemplo de código, se muestra cómo usar `ListRecipes`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void listAllRecipes(PersonalizeClient personalizeClient) {  
  
    try {  
        ListRecipesRequest recipesRequest = ListRecipesRequest.builder()  
            .maxResults(15)  
            .build();  
  
        ListRecipesResponse response =  
personalizeClient.listRecipes(recipesRequest);  
        List<RecipeSummary> recipes = response.recipes();  
        for (RecipeSummary recipe : recipes) {  
            System.out.println("The recipe ARN is: " + recipe.recipeArn());  
            System.out.println("The recipe name is: " + recipe.name());  
        }  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
    }  
}
```

- Para obtener más información sobre la API, consulta [ListRecipes](#) la Referencia AWS SDK for Java 2.x de la API.

ListSolutions

En el siguiente ejemplo de código, se muestra cómo usar `ListSolutions`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void listAllSolutions(PersonalizeClient personalizeClient, String  
datasetGroupArn) {  
  
    try {  
        ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()  
            .maxResults(10)  
            .datasetGroupArn(datasetGroupArn)  
            .build();  
  
        ListSolutionsResponse response =  
personalizeClient.listSolutions(solutionsRequest);  
        List<SolutionSummary> solutions = response.solutions();  
        for (SolutionSummary solution : solutions) {  
            System.out.println("The solution ARN is: " +  
solution.solutionArn());  
            System.out.println("The solution name is: " + solution.name());  
        }  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
}
```

- Para obtener más información sobre la API, consulta [ListSolutions](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateCampaign

En el siguiente ejemplo de código, se muestra cómo usar UpdateCampaign.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String updateCampaign(PersonalizeClient personalizeClient,
    String campaignArn,
    String solutionVersionArn,
    Integer minProvisionedTPS) {

    try {
        // build the updateCampaignRequest
        UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
            .campaignArn(campaignArn)
            .solutionVersionArn(solutionVersionArn)
            .minProvisionedTPS(minProvisionedTPS)
            .build();

        // update the campaign
        personalizeClient.updateCampaign(updateCampaignRequest);

        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();
```

```
DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
Campaign updatedCampaign = campaignResponse.campaign();

System.out.println("The Campaign status is " +
updatedCampaign.status());
return updatedCampaign.status();

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

- Para obtener más información sobre la API, consulta [UpdateCampaign](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de eventos de Amazon Personalize usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante Amazon Personalize Events. AWS SDK for Java 2.x

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

PutEvents

En el siguiente ejemplo de código, se muestra cómo usar PutEvents.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String item1Id,
    String item1PropertyName,
    String item1PropertyValue,
    String item2Id,
    String item2PropertyName,
    String item2PropertyValue) {

    int responseCode = 0;
    ArrayList<Item> items = new ArrayList<>();

    try {
        Item item1 = Item.builder()
            .itemId(item1Id)
            .properties(String.format("{\\"%1$s\\": \\"%2$s
\}"),
                item1PropertyName,
                item1PropertyValue))
            .build();

        items.add(item1);

        Item item2 = Item.builder()
            .itemId(item2Id)
            .properties(String.format("{\\"%1$s\\": \\"%2$s
\}"),
                item2PropertyName,
                item2PropertyValue))
```

```
                .build();

        items.add(item2);

        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
                .datasetArn(datasetArn)
                .items(items)
                .build();

        responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}
```

- Para obtener más información sobre la API, consulta [PutEvents](#) la Referencia AWS SDK for Java 2.x de la API.

PutUsers

En el siguiente ejemplo de código, se muestra cómo usar PutUsers.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
        String datasetArn,
        String user1Id,
        String user1PropertyName,
        String user1PropertyValue,
```

```
        String user2Id,
        String user2PropertyName,
        String user2PropertyValue) {

    int responseCode = 0;
    ArrayList<User> users = new ArrayList<>();

    try {
        User user1 = User.builder()
            .userId(user1Id)
            .properties(String.format("{\\"%1$s\\": \\"%2$s
\\"}",
            user1PropertyName,
            user1PropertyValue))
            .build();

        users.add(user1);

        User user2 = User.builder()
            .userId(user2Id)
            .properties(String.format("{\\"%1$s\\": \\"%2$s
\\"}",
            user2PropertyName,
            user2PropertyValue))
            .build();

        users.add(user2);

        PutUsersRequest putUsersRequest = PutUsersRequest.builder()
            .datasetArn(datasetArn)
            .users(users)
            .build();

        responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}
```


- Para obtener más información sobre la API, consulta [PutUsers](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Amazon Personalize Runtime usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante Amazon Personalize Runtime. AWS SDK for Java 2.x

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

GetPersonalizedRanking

En el siguiente ejemplo de código, se muestra cómo usar `GetPersonalizedRanking`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
```

```
String campaignArn,
String userId,
ArrayList<String> items) {

    try {
        GetPersonalizedRankingRequest rankingRecommendationsRequest =
GetPersonalizedRankingRequest.builder()
            .campaignArn(campaignArn)
            .userId(userId)
            .inputList(items)
            .build();

        GetPersonalizedRankingResponse recommendationsResponse =
personalizeRuntimeClient
            .getPersonalizedRanking(rankingRecommendationsRequest);
        List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
        int rank = 1;
        for (PredictedItem item : rankedItems) {
            System.out.println("Item ranked at position " + rank + " details");
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
            System.out.println("-----");
            rank++;
        }
        return rankedItems;
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```

- Para obtener más información sobre la API, consulta [GetPersonalizedRanking](#) la Referencia AWS SDK for Java 2.x de la API.

GetRecommendations

En el siguiente ejemplo de código, se muestra cómo usar `GetRecommendations`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtener una lista de artículos 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);
    }
}
```

Obtenga una lista de recomendaciones de un recomendador creado en un grupo de conjuntos de datos de dominio.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn,
```

```
String userId) {

    try {
        GetRecommendationsRequest recommendationsRequest =
        GetRecommendationsRequest.builder()
            .recommenderArn(recommenderArn)
            .numResults(20)
            .userId(userId)
            .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
            .getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item : items) {
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
        }
    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Use un filtro cuando solicite recomendaciones.

```
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 obtener más información sobre la API, consulta [GetRecommendations](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Amazon Pinpoint usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Amazon Pinpoint.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

CreateApp

En el siguiente ejemplo de código, se muestra cómo usar CreateApp.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateApp](#) la Referencia AWS SDK for Java 2.x de la API.

CreateCampaign

En el siguiente ejemplo de código, se muestra cómo usar CreateCampaign.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear una campaña.

```
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 obtener más información sobre la API, consulta [CreateCampaign](#) la Referencia AWS SDK for Java 2.x de la API.

CreateExportJob

En el siguiente ejemplo de código, se muestra cómo usar `CreateExportJob`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Exportar un punto de conexión.

```
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 obtener más información sobre la API, consulta [CreateExportJob](#) la Referencia AWS SDK for Java 2.x de la API.

CreateImportJob

En el siguiente ejemplo de código, se muestra cómo usar `CreateImportJob`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Importar un 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 obtener más información sobre la API, consulta [CreateImportJob](#) la Referencia AWS SDK for Java 2.x de la API.

CreateSegment

En el siguiente ejemplo de código, se muestra cómo usar CreateSegment.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateSegment](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteApp

En el siguiente ejemplo de código, se muestra cómo usar DeleteApp.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine una aplicación.

```
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 obtener más información sobre la API, consulta [DeleteApp](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteEndpoint

En el siguiente ejemplo de código, se muestra cómo usar DeleteEndpoint.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Eliminar un punto de conexión.

```
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();

deletePinEncpoint(pinpoint, appId, endpointId);
pinpoint.close();
}

public static void deletePinEncpoint(PinpointClient pinpoint, String appId,
String endpointId) {
    try {
        DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .build();

        DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
        String id = result.endpointResponse().id();
        System.out.println("The deleted endpoint id  " + id);

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- Para obtener más información sobre la API, consulta [DeleteEndpoint](#) la Referencia AWS SDK for Java 2.x de la API.

GetEndpoint

En el siguiente ejemplo de código, se muestra cómo usar GetEndpoint.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetEndpoint](#) la Referencia AWS SDK for Java 2.x de la API.

GetSegments

En el siguiente ejemplo de código, se muestra cómo usar `GetSegments`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Enumerar 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 obtener más información sobre la API, consulta [GetSegments](#) la Referencia AWS SDK for Java 2.x de la API.

GetSmsChannel

En el siguiente ejemplo de código, se muestra cómo usar `GetSmsChannel`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetSmsChannel](#) en la Referencia AWS SDK for Java 2.x de la API.

GetUserEndpoints

En el siguiente ejemplo de código, se muestra cómo usar GetUserEndpoints.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetUserEndpoints](#) la Referencia AWS SDK for Java 2.x de la API.

SendMessage

En el siguiente ejemplo de código, se muestra cómo usar SendMessage.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Enviar un mensaje de correo electrónico.

```
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);
    }
}
}
}

```

Envíe un mensaje de correo electrónico con valores de CC.

```

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 SendMessageCC {

    // 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();
    }
}
```

```
}  
}
```

Enviar un mensaje 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.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 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.getMessageResponse();
        Map map1 = msg1.getResult();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.getAwsErrorDetails().getErrorMessage());
        System.exit(1);
    }
```

```
    }  
  }  
}
```

Enviar mensajes SMS por lotes.

```
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.sendMessages(request);
```

```
        MessageResponse msg1 = response.messageResponse();
        Map map1 = msg1.result();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

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

- Para obtener más información sobre la API, consulta [SendMessages](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateEndpoint

En el siguiente ejemplo de código, se muestra cómo usar UpdateEndpoint.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [UpdateEndpoint](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de código de la API de SMS y voz de Amazon Pinpoint usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante la AWS SDK for Java 2.x API de SMS y voz de Amazon Pinpoint.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

SendVoiceMessage

En el siguiente ejemplo de código, se muestra cómo usar `SendVoiceMessage`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [SendVoiceMessage](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Amazon Polly usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Amazon Polly.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

DescribeVoices

En el siguiente ejemplo de código, se muestra cómo usar DescribeVoices.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeVoices](#) la Referencia AWS SDK for Java 2.x de la API.

ListLexicons

En el siguiente ejemplo de código, se muestra cómo usar `ListLexicons`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListLexicons](#) la Referencia AWS SDK for Java 2.x de la API.

SynthesizeSpeech

En el siguiente ejemplo de código, se muestra cómo usar SynthesizeSpeech.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [SynthesizeSpeech](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Amazon RDS usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Amazon RDS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Amazon RDS

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon RDS.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener información sobre la API, consulte [DescribeDBInstances](#) en la Referencia de la API de AWS SDK for Java 2.x .

Temas

- [Acciones](#)
- [Escenarios](#)
- [Ejemplos sin servidor](#)

Acciones

CreateDBInstance

En el siguiente ejemplo de código, se muestra cómo usar CreateDBInstance.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener información sobre la API, consulte [CreateDBInstance](#) en la Referencia de la API de AWS SDK for Java 2.x .

CreateDBParameterGroup

En el siguiente ejemplo de código, se muestra cómo usar CreateDBParameterGroup.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
```

```
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

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

- Para obtener más información sobre la API, consulta [CreateDB ParameterGroup](#) en la referencia de la AWS SDK for Java 2.x API.

CreateDBSnapshot

En el siguiente ejemplo de código, se muestra cómo usar CreateDBSnapshot.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
```

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

- Para obtener información sobre la API, consulte [CreateDBSnapshot](#) en la Referencia de la API de AWS SDK for Java 2.x .

DeleteDBInstance

En el siguiente ejemplo de código, se muestra cómo usar DeleteDBInstance.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener información sobre la API, consulte [DeleteDBInstance](#) en la Referencia de la API de AWS SDK for Java 2.x .

DeleteDBParameterGroup

En el siguiente ejemplo de código, se muestra cómo usar DeleteDBParameterGroup.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while database
// exists.
public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
```

```
        // Went through the entire list and did not find the
database ARN.
        isDataDel = true;
    }
    Thread.sleep(sleepTime * 1000);
    index++;
}

// Delete the para group.
DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
    .dbParameterGroupName(dbGroupName)
    .build();

rdsClient.deleteDBParameterGroup(parameterGroupRequest);
System.out.println(dbGroupName + " was deleted.");

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

- Para obtener más información sobre la API, consulta [DeleteDB ParameterGroup en la referencia](#) de la AWS SDK for Java 2.x API.

DescribeAccountAttributes

En el siguiente ejemplo de código, se muestra cómo usar DescribeAccountAttributes.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeAccountAttributes](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeDBEngineVersions

En el siguiente ejemplo de código, se muestra cómo usar DescribeDBEngineVersions.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

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

        // Get all DBEngineVersion objects.
        for (DBEngineVersion 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 obtener información detallada sobre la API, consulta [DescribeDB EngineVersions en la referencia de la AWS SDK for Java 2.x API](#).

DescribeDBInstances

En el siguiente ejemplo de código, se muestra cómo usar DescribeDBInstances.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener información sobre la API, consulte [DescribeDBInstances](#) en la Referencia de la API de AWS SDK for Java 2.x .

DescribeDBParameterGroups

En el siguiente ejemplo de código, se muestra cómo usar DescribeDBParameterGroups.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
```

```
try {
    DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
    .dbParameterGroupName(dbGroupName)
    .maxRecords(20)
    .build();

    DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
    List<DBParameterGroup> groups = response.dbParameterGroups();
    for (DBParameterGroup group : groups) {
        System.out.println("The group name is " +
group.dbParameterGroupName());
        System.out.println("The group description is " +
group.description());
    }

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

- Para obtener información detallada sobre la API, consulta [DescribeDB ParameterGroups en la referencia de la AWS SDK for Java 2.x API](#).

DescribeDBParameters

En el siguiente ejemplo de código, se muestra cómo usar DescribeDBParameters.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Retrieve parameters in the group.
```

```

    public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .source("user")
                    .build();
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                    || (paraName.compareTo("auto_increment_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 obtener información sobre la API, consulte [DescribeDBParameters](#) en la Referencia de la API de AWS SDK for Java 2.x .

DescribeOrderableDBInstanceOptions

En el siguiente ejemplo de código, se muestra cómo usar `DescribeOrderableDBInstanceOptions`.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

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

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

- Para obtener más información sobre la API, consulte la [DescribeOrderablebase de datos InstanceOptions](#) en AWS SDK for Java 2.x la referencia de API.

GenerateRDSAuthToken

En el siguiente ejemplo de código, se muestra cómo usar GenerateRDSAuthToken.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa la [RdsUtilities](#) clase para generar un token de autenticación.

```
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 obtener más información sobre la API, consulte [GenerateRDS AuthToken en la referencia de la AWS SDK for Java 2.x API](#).

ModifyDBInstance

En el siguiente ejemplo de código, se muestra cómo usar ModifyDBInstance.

SDK para Java 2.x

 Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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.println("The ARN of the modified database is: " +
instanceResponse.dbInstance().dbInstanceArn());

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

- Para obtener información sobre la API, consulte [ModifyDBInstance](#) en la Referencia de la API de AWS SDK for Java 2.x .

ModifyDBParameterGroup

En el siguiente ejemplo de código, se muestra cómo usar `ModifyDBParameterGroup`.

SDK para Java 2.x

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// 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 obtener más información sobre la API, consulta [ModifyDB ParameterGroup](#) en la referencia de la AWS SDK for Java 2.x API.

RebootDBInstance

En el siguiente ejemplo de código, se muestra cómo usar `RebootDBInstance`.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener información sobre la API, consulte [RebootDBInstance](#) en la Referencia de la API de AWS SDK for Java 2.x .

Escenarios

Comience a utilizar instancias de base de datos

En el siguiente ejemplo de código, se muestra cómo:

- Cree un grupo de parámetros de base de datos personalizado y defina los valores de los parámetros.

- Cree una instancia de base de datos que esté configurada para utilizar el grupo de parámetros. La instancia de base de datos también contiene una base de datos.
- Cree una instantánea de la instancia.
- Elimine la instancia y el grupo de parámetros.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute varias operaciones.

```
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.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

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

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

            DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
                .dbSnapshotIdentifier(dbSnapshotIdentifier)
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .build();

            while (!snapshotReady) {
                DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
                List<DBSnapshot> snapshotList = response.dbSnapshots();
                for (DBSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");
        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
        }
    }
}
```

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

// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

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

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

        String endpoint = "";
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
            }
        }
    }
}
```

```
        if (instanceReadyStr.contains("available")) {
            endpoint = instance.endpoint().address();
            instanceReady = true;
        } else {
            System.out.print(".");
            Thread.sleep(sleepTime * 1000);
        }
    }
}

System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

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

// Create a database instance and return the ARN of the database.
public static String createDatabaseInstance(RdsClient rdsClient,
    String dbGroupName,
    String dbInstanceIdentifier,
    String dbName,
    String masterUsername,
    String masterUserPassword) {

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .allocatedStorage(100)
            .dbName(dbName)
            .dbParameterGroupName(dbGroupName)
            .engine("mysql")
            .dbInstanceClass("db.m4.large")
            .engineVersion("8.0")
            .storageType("standard")
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
```

```
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

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

    return "";
}

// Get a list of micro instances.
public static void getMicroInstances(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
            .builder()
            .engine("mysql")
            .build();

        DescribeOrderableDbInstanceOptionsResponse response = rdsClient
            .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
        List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
        for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
        {
            System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
            System.out.println("The engine description is " +
dbInstanceOption.engine());
        }

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

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
```



```
DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .dbParameterGroupFamily(dbParameterGroupFamily)
    .engine("mysql")
    .build();

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

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

// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");
    }
}
```

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

// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }

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

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

    public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .maxRecords(20)
                .build();

            DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbParameterGroupName());
                System.out.println("The group description is " +
group.description());
            }

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

    public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
        try {
            CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .dbParameterGroupFamily(dbParameterGroupFamily)
                .description("Created by using the AWS SDK for Java")
                .build();

            CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
```

```
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

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

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

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

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

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

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .

- [CreateDBInstance](#)
- [Creó a B. ParameterGroup](#)
- [CreateDBSnapshot](#)
- [DeleteDBInstance](#)
- [Eliminado B ParameterGroup](#)
- [Descrito B EngineVersions](#)
- [DescribeDBInstances](#)
- [Descrito B ParameterGroups](#)
- [DescribeDBParameters](#)
- [DescribeDBSnapshots](#)
- [DescribeOrderableDB InstanceOptions](#)
- [Modificar DB ParameterGroup](#)

Ejemplos sin servidor

Conexión a una base de datos de Amazon RDS en una función de Lambda

En el siguiente ejemplo de código, se muestra cómo implementar una función de Lambda que se conecta a una base de datos de RDS. La función realiza una solicitud sencilla a la base de datos y devuelve el resultado.

SDK para Java 2.x

Note

Hay más en marcha. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Conexión a una base de datos de Amazon RDS en una función Lambda mediante Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements RequestHandler<APIGatewayProxyRequestEvent,
APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
event, Context context) {
        APIGatewayProxyResponseEvent response = new APIGatewayProxyResponseEvent();

        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                System.getenv("ProxyHostName"),
                System.getenv("Port"),
                System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
DriverManager.getConnection(connectionString, System.getenv("DBUserName"), token);
                PreparedStatement statement = connection.prepareStatement("SELECT ?
+ ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }
        }
    }
}
```

```
        }
    }

    } catch (Exception e) {
        response.setStatusCode(500);
        response.setBody("Error: " + e.getMessage());
    }

    return response;
}

private String createAuthToken() {
    // Create RDS Data Service client
    RdsDataClient rdsDataClient = RdsDataClient.builder()
        .region(Region.of(System.getenv("AWS_REGION")))
        .credentialsProvider(DefaultCredentialsProvider.create())
        .build();

    // Define authentication request
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .resourceArn(System.getenv("ProxyHostName"))
        .secretArn(System.getenv("DBUserName"))
        .database(System.getenv("DBName"))
        .sql("SELECT 'RDS IAM Authentication'")
        .build();

    // Execute request and obtain authentication token
    ExecuteStatementResponse response = rdsDataClient.executeStatement(request);
    Field tokenField = response.records().get(0).get(0);

    return tokenField.stringValue();
}
}
```

Ejemplos de Amazon Redshift usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Amazon Redshift.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Hello Amazon Redshift

En los siguientes ejemplos de código, se muestra cómo empezar a utilizar Amazon Redshift.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.paginators.DescribeClustersIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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 HelloRedshift {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RedshiftClient redshiftClient = RedshiftClient.builder()
```



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

    listClustersPaginator(redshiftClient);
}

public static void listClustersPaginator(RedshiftClient redshiftClient) {
    DescribeClustersIterable clustersIterable =
redshiftClient.describeClustersPaginator();
    clustersIterable.stream()
        .flatMap(r -> r.clusters().stream())
        .forEach(cluster -> System.out
            .println(" Cluster identifier: " + cluster.clusterIdentifier() + "
status = " + cluster.clusterStatus()));
}
}
```

- Para obtener más información sobre la API, consulte [describeClusters](#) en la Referencia de la API de AWS SDK for Java 2.x .

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateCluster

En el siguiente ejemplo de código, se muestra cómo usar CreateCluster.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree el clúster.

```
public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
                                String masterUserPassword) {
    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .nodeType("ra3.4xlarge")
            .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 obtener más información sobre la API, consulta [CreateCluster](#) la Referencia AWS SDK for Java 2.x de la API.

CreateTable

En el siguiente ejemplo de código, se muestra cómo usar CreateTable.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void createTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName) {
    try {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
        .clusterIdentifier(clusterId)
        .dbUser(userName)
        .database(databaseName)
        .sql("CREATE TABLE Movies ("
            + "id INT PRIMARY KEY, "
            + "title VARCHAR(100), "
            + "year INT)")
        .build();

        redshiftDataClient.executeStatement(createTableRequest);
        System.out.println("Table created: Movies");

    } catch (RedshiftDataException e) {
        System.err.println("Error creating table: " + e.getMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [CreateTable](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteCluster

En el siguiente ejemplo de código, se muestra cómo usar DeleteCluster.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Eliminar el clúster.

```
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 obtener más información sobre la API, consulta [DeleteCluster](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeClusters

En el siguiente ejemplo de código, se muestra cómo usar DescribeClusters.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Describir el clúster.

```
public static void waitForClusterReady(RedshiftClient redshiftClient, String
clusterId) {
    boolean clusterReady = false;
```

```
String clusterReadyStr;
System.out.println("Waiting for cluster to become available. This may take a
few mins.");
try {
    DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
        .clusterIdentifier(clusterId)
        .build();
    long startTime = System.currentTimeMillis();

    // Loop until the cluster is ready.
    while (!clusterReady) {
        DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters(clustersRequest);
        List<Cluster> clusterList = clusterResponse.clusters();
        for (Cluster cluster : clusterList) {
            clusterReadyStr = cluster.clusterStatus();
            if (clusterReadyStr.contains("available"))
                clusterReady = true;
            else {
                long elapsedTimeMillis = System.currentTimeMillis() -
startTime;

                long elapsedSeconds = elapsedTimeMillis / 1000;
                long minutes = elapsedSeconds / 60;
                long seconds = elapsedSeconds % 60;

                System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
                TimeUnit.SECONDS.sleep(5);
            }
        }
    }

    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
    long elapsedSeconds = elapsedTimeMillis / 1000;
    long minutes = elapsedSeconds / 60;
    long seconds = elapsedSeconds % 60;

    System.out.println(String.format("Cluster is available! Total Elapsed
Time: %02d:%02d", minutes, seconds));

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

```
    }  
}
```

- Para obtener más información sobre la API, consulta [DescribeClusters](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeStatement

En el siguiente ejemplo de código, se muestra cómo usar DescribeStatement.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void checkStatement(RedshiftDataClient redshiftDataClient, String  
sqlId) {  
    try {  
        DescribeStatementRequest statementRequest =  
DescribeStatementRequest.builder()  
            .id(sqlId)  
            .build();  
  
        String status;  
        while (true) {  
            DescribeStatementResponse response =  
redshiftDataClient.describeStatement(statementRequest);  
            status = response.statusAsString();  
            System.out.println("..." + status);  
  
            if (status.compareTo("FAILED") == 0 ) {  
                System.out.println("The Query Failed. Ending program");  
                System.exit(1);  
            } else if (status.compareTo("FINISHED") == 0) {  
                break;  
            }  
        }  
    }  
}
```

```
        TimeUnit.SECONDS.sleep(1);
    }

    System.out.println("The statement is finished!");

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

- Para obtener más información sobre la API, consulta [DescribeStatement](#) la Referencia AWS SDK for Java 2.x de la API.

GetStatementResult

En el siguiente ejemplo de código, se muestra cómo usar `GetStatementResult`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Compruebe el resultado de la instrucción.

```
public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
    try {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
        .id(statementId)
        .build();

        // Extract and print the field values using streams.
        GetStatementResultResponse response =
redshiftDataClient.getStatementResult(resultRequest);
        response.records().stream()
            .flatMap(List::stream)
```

```
        .map(Field::stringValue)
        .filter(value -> value != null)
        .forEach(value -> System.out.println("The Movie title field is " +
value));

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

- Para obtener más información sobre la API, consulta [GetStatementResult](#) la Referencia AWS SDK for Java 2.x de la API.

Insert

En el siguiente ejemplo de código, se muestra cómo usar Insert.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void popTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName, String fileName, int number)
throws IOException {
    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()) {
        if (t == number)
            break;
        currentNode = (ObjectNode) iter.next();
        int year = currentNode.get("year").asInt();
```



```
String title = currentNode.get("title").asText();

// Use SqlParameter to avoid SQL injection.
List<SqlParameter> parameterList = new ArrayList<>();
String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";

// Create the parameters.
SqlParameter idParam = SqlParameter.builder()
    .name("id")
    .value(String.valueOf(t))
    .build();

SqlParameter titleParam= SqlParameter.builder()
    .name("title")
    .value(title)
    .build();

SqlParameter yearParam = SqlParameter.builder()
    .name("year")
    .value(String.valueOf(year))
    .build();
parameterList.add(idParam);
parameterList.add(titleParam);
parameterList.add(yearParam);

try {
    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
        .clusterIdentifier(clusterId)
        .sql(sqlStatement)
        .database(databaseName)
        .dbUser(userName)
        .parameters(parameterList)
        .build();

    redshiftDataClient.executeStatement(insertStatementRequest);
    System.out.println("Inserted: " + title + " (" + year + ")");
    t++;

} catch (RedshiftDataException e) {
    System.err.println("Error inserting data: " + e.getMessage());
    System.exit(1);
}
```

```
    }  
    System.out.println(t + " records were added to the Movies table. ");  
}
```

- Para obtener más información acerca de la API, consulte [Insert](#) en la Referencia de la API de AWS SDK for Java 2.x .

ModifyCluster

En el siguiente ejemplo de código, se muestra cómo usar `ModifyCluster`.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Modificar un clúster

```
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 obtener más información sobre la API, consulta [ModifyCluster](#) la Referencia AWS SDK for Java 2.x de la API.

Query

En el siguiente ejemplo de código, se muestra cómo usar Query.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Consulte una tabla.

```
public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
                                       String database,
                                       String dbUser,
                                       int year,
                                       String clusterId) {

    try {
        String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam= SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();
```

```
ExecuteStatementResponse response =
redshiftDataClient.executeStatement(statementRequest);
return response.id();

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

- Para obtener información sobre la API, consulte [Query](#) en la referencia de la API de AWS SDK for Java 2.x .

Escenarios

Introducción a Amazon Redshift

En el siguiente ejemplo de código, se muestra cómo trabajar con tablas, elementos y consultas de Amazon Redshift.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ObjectNode;
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.CreateClusterRequest;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterRequest;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
```

```
import software.amazon.awssdk.services.redshift.model.DescribeClustersRequest;
import software.amazon.awssdk.services.redshift.model.DescribeClustersResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterRequest;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import software.amazon.awssdk.services.redshiftdata.RedshiftDataClient;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.Field;
import software.amazon.awssdk.services.redshiftdata.model.GetStatementResultRequest;
import
    software.amazon.awssdk.services.redshiftdata.model.GetStatementResultResponse;
import software.amazon.awssdk.services.redshiftdata.model.ListDatabasesRequest;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import software.amazon.awssdk.services.redshiftdata.model.SqlParameter;
import
    software.amazon.awssdk.services.redshiftdata.paginators.ListDatabasesIterable;
import com.fasterxml.jackson.core.JsonParser;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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. Prompts the user for a unique cluster ID or use the default value.
 * 2. Creates a Redshift cluster with the specified or default cluster Id value.
 * 3. Waits until the Redshift cluster is available for use.
 * 4. Lists all databases using a pagination API call.
 * 5. Creates a table named "Movies" with fields ID, title, and year.
```

- * 6. Inserts a specified number of records into the "Movies" table by reading the Movies JSON file.
 - * 7. Prompts the user for a movie release year.
 - * 8. Runs a SQL query to retrieve movies released in the specified year.
 - * 9. Modifies the Redshift cluster.
 - * 10. Prompts the user for confirmation to delete the Redshift cluster.
 - * 11. If confirmed, deletes the specified Redshift cluster.
- */

```
public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <jsonFilePath>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate that
file in ../../../../resources/sample_files/movies.json)
                """;

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

        String jsonFilePath = args[0];
        String userName;
        String userPassword;
        String databaseName = "dev" ;
        Scanner scanner = new Scanner(System.in);

        Region region = Region.US_EAST_1;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        RedshiftDataClient redshiftDataClient = RedshiftDataClient.builder()
            .region(region)
            .build();

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

```
System.out.println("Welcome to the Amazon Redshift SDK Getting Started
scenario.");
System.out.println("""
This Java program demonstrates how to interact with Amazon Redshift by using
the AWS SDK for Java (v2).\s
Amazon Redshift is a fully managed, petabyte-scale data warehouse service
hosted in the cloud.

The program's primary functionalities include cluster creation, verification
of cluster readiness,\s
list databases, table creation, data population within the table, and
execution of SQL statements.
Furthermore, it demonstrates the process of querying data from the Movie
table.\s

Upon completion of the program, all AWS resources are cleaned up.
""");

System.out.println("Lets get started...");
System.out.println("Please enter your user name (default is awsuser)");
String user = scanner.nextLine();
userName = user.isEmpty() ? "awsuser" : user;
System.out.println(DASHES);
System.out.println("Please enter your user password (default is
AwsUser1000)");
String userpass = scanner.nextLine();
userPassword = userpass.isEmpty() ? "AwsUser1000" : userpass;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("A Redshift cluster refers to the collection of computing
resources and storage that work together to process and analyze large volumes of
data.");
System.out.println("Enter a cluster id value (default is redshift-cluster-
movies): ");
String userClusterId = scanner.nextLine();
String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
userClusterId;
createCluster(redshiftClient, clusterId, userName, userPassword);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Wait until "+clusterId +" is available.");
System.out.print("Press Enter to continue...");
```

```
scanner.nextLine();
waitForClusterReady(redshiftClient, clusterId);
System.out.println(DASHES);

System.out.println(DASHES);
String databaseInfo = ""
    When you created $clusteridD, the dev database is created by default and
used in this scenario.\s

    To create a custom database, you need to have a CREATEDB privilege.\s
    For more information, see the documentation here: https://
docs.aws.amazon.com/redshift/latest/dg/r\_CREATE\_DATABASE.html.
    """.replace("$clusteridD", clusterId);

System.out.println(databaseInfo);
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("List databases in "+clusterId);
System.out.print("Press Enter to continue...");
scanner.nextLine();
listAllDatabases(redshiftDataClient, clusterId, userName, databaseName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Now you will create a table named Movies.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
createTable(redshiftDataClient, clusterId, databaseName, userName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Populate the Movies table using the Movies.json file.");
System.out.println("Specify the number of records you would like to add to
the Movies Table.");
System.out.println("Please enter a value between 50 and 200.");
int numRecords;
do {
    System.out.print("Enter a value: ");
    while (!scanner.hasNextInt()) {
        System.out.println("Invalid input. Please enter a value between 50
and 200.");
```



```
        System.out.print("Enter a year: ");
        scanner.next();
    }
    numRecords = scanner.nextInt();
} while (numRecords < 50 || numRecords > 200);
popTable(redshiftDataClient, clusterId, databaseName, userName,
jsonFilePath, numRecords);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Query the Movies table by year. Enter a value between
2012-2014.");
int movieYear;
do {
    System.out.print("Enter a year: ");
    while (!scanner.hasNextInt()) {
        System.out.println("Invalid input. Please enter a valid year between
2012 and 2014.");
        System.out.print("Enter a year: ");
        scanner.next();
    }
    movieYear = scanner.nextInt();
    scanner.nextLine();
} while (movieYear < 2012 || movieYear > 2014);

String id = queryMoviesByYear(redshiftDataClient, databaseName, userName,
movieYear, clusterId);
System.out.println("The identifier of the statement is " + id);
checkStatement(redshiftDataClient, id);
getResults(redshiftDataClient, id);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Now you will modify the Redshift cluster.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
modifyCluster(redshiftClient, clusterId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Would you like to delete the Amazon Redshift cluster?
(y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
```

```
        System.out.println("You selected to delete " +clusterId);
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        deleteRedshiftCluster(redshiftClient, clusterId);
    } else {
        System.out.println("The "+clusterId +" was not deleted");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("This concludes the Amazon Redshift SDK Getting Started
scenario.");
    System.out.println(DASHES);
}

public static void listAllDatabases(RedshiftDataClient redshiftDataClient,
String clusterId, String dbUser, String database) {
    try {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        ListDatabasesIterable listDatabasesIterable =
redshiftDataClient.listDatabasesPaginator(databasesRequest);
        listDatabasesIterable.stream()
            .flatMap(r -> r.databases().stream())
            .forEach(db -> System.out
                .println("The database name is : " + db));

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

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);
    }
}

public static void popTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName, String fileName, int number)
throws IOException {
    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()) {
        if (t == number)
            break;
        currentNode = (ObjectNode) iter.next();
        int year = currentNode.get("year").asInt();
        String title = currentNode.get("title").asText();

        // Use SqlParameter to avoid SQL injection.
        List<SqlParameter> parameterList = new ArrayList<>();
        String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";

        // Create the parameters.
        SqlParameter idParam = SqlParameter.builder()
            .name("id")
            .value(String.valueOf(t))
            .build();

        SqlParameter titleParam= SqlParameter.builder()
            .name("title")
            .value(title)
            .build();
```

```
        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();
        parameterList.add(idParam);
        parameterList.add(titleParam);
        parameterList.add(yearParam);

        try {
            ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                .clusterIdentifier(clusterId)
                .sql(sqlStatement)
                .database(databaseName)
                .dbUser(userName)
                .parameters(parameterList)
                .build();

            redshiftDataClient.executeStatement(insertStatementRequest);
            System.out.println("Inserted: " + title + " (" + year + ")");
            t++;

        } catch (RedshiftDataException e) {
            System.err.println("Error inserting data: " + e.getMessage());
            System.exit(1);
        }
    }
    System.out.println(t + " records were added to the Movies table. ");
}

public static void checkStatement(RedshiftDataClient redshiftDataClient, String
sqlId) {
    try {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        String status;
        while (true) {
            DescribeStatementResponse response =
redshiftDataClient.describeStatement(statementRequest);
            status = response.statusAsString();
        }
    }
}
```

```
        System.out.println("..." + status);

        if (status.compareTo("FAILED") == 0 ) {
            System.out.println("The Query Failed. Ending program");
            System.exit(1);

        } else if (status.compareTo("FINISHED") == 0) {
            break;
        }
        TimeUnit.SECONDS.sleep(1);
    }

    System.out.println("The statement is finished!");

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

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);
    }
}

public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
String database,
```

```
        String dbUser,
        int year,
        String clusterId) {

    try {
        String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam= SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        ExecuteStatementResponse response =
redshiftDataClient.executeStatement(statementRequest);
        return response.id();

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

public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
    try {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        // Extract and print the field values using streams.
        GetStatementResultResponse response =
redshiftDataClient.getStatementResult(resultRequest);
        response.records().stream()
            .flatMap(List::stream)
```

```
        .map(Field::stringValue)
        .filter(value -> value != null)
        .forEach(value -> System.out.println("The Movie title field is " +
value));

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

public static void waitForClusterReady(RedshiftClient redshiftClient, String
clusterId) {
    boolean clusterReady = false;
    String clusterReadyStr;
    System.out.println("Waiting for cluster to become available. This may take a
few mins.");
    try {
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();
        long startTime = System.currentTimeMillis();

        // Loop until the cluster is ready.
        while (!clusterReady) {
            DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters(clustersRequest);
            List<Cluster> clusterList = clusterResponse.clusters();
            for (Cluster cluster : clusterList) {
                clusterReadyStr = cluster.clusterStatus();
                if (clusterReadyStr.contains("available"))
                    clusterReady = true;
                else {
                    long elapsedTimeMillis = System.currentTimeMillis() -
startTime;

                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;

                    System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
                    TimeUnit.SECONDS.sleep(5);
                }
            }
        }
    }
}
```

```
    }
  }

  long elapsedTimeMillis = System.currentTimeMillis() - startTime;
  long elapsedSeconds = elapsedTimeMillis / 1000;
  long minutes = elapsedSeconds / 60;
  long seconds = elapsedSeconds % 60;

  System.out.println(String.format("Cluster is available! Total Elapsed
Time: %02d:%02d", minutes, seconds));

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

public static void createTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName) {
  try {
    ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
      .clusterIdentifier(clusterId)
      .dbUser(userName)
      .database(databaseName)
      .sql("CREATE TABLE Movies ("
        + "id INT PRIMARY KEY, "
        + "title VARCHAR(100), "
        + "year INT)")
      .build();

    redshiftDataClient.executeStatement(createTableRequest);
    System.out.println("Table created: Movies");

  } catch (RedshiftDataException e) {
    System.err.println("Error creating table: " + e.getMessage());
    System.exit(1);
  }
}

public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
                                String masterUserPassword) {
  try {
```



```
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .nodeType("ra3.4xlarge")
            .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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [createCluster](#)
 - [describeClusters](#)
 - [describeStatement](#)
 - [executeStatement](#)
 - [getStatementResult](#)
 - [listDatabasesPaginator](#)
 - [modifyCluster](#)

Ejemplos de Amazon Rekognition usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes AWS SDK for Java 2.x con Amazon Rekognition.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CompareFaces

En el siguiente ejemplo de código, se muestra cómo usar CompareFaces.

Para obtener información, consulte [Comparación de rostros en imágenes](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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.InputStream;
```

```
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> uncomparing = compareFacesResult.unmatchedFaces();
    System.out.println("There was " + uncomparing.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 obtener más información sobre la API, consulta [CompareFaces](#) la Referencia AWS SDK for Java 2.x de la API.

CreateCollection

En el siguiente ejemplo de código, se muestra cómo usar CreateCollection.

Para obtener información, consulte [Creación de una colección](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateCollection](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteCollection

En el siguiente ejemplo de código, se muestra cómo usar DeleteCollection.

Para obtener información, consulte [Eliminación de una colección](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteCollection](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteFaces

En el siguiente ejemplo de código, se muestra cómo usar DeleteFaces.

Para obtener información, consulte [Eliminación de rostros de una colección](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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(RekognitionClient rekClient,
    String collectionId,
    String faceId) {

    try {
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Para obtener más información sobre la API, consulta [DeleteFaces](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeCollection

En el siguiente ejemplo de código, se muestra cómo usar DescribeCollection.

Para obtener información, consulte [Descripción de una colección](#).

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeCollection](#) la Referencia AWS SDK for Java 2.x de la API.

DetectFaces

En el siguiente ejemplo de código, se muestra cómo usar DetectFaces.

Para obtener información, consulte [Detección de rostros en una imagen](#).

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DetectFaces](#) la Referencia AWS SDK for Java 2.x de la API.

DetectLabels

En el siguiente ejemplo de código, se muestra cómo usar DetectLabels.

Para obtener información, consulte [Detección de etiquetas en una imagen](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DetectLabels](#) la Referencia AWS SDK for Java 2.x de la API.

DetectModerationLabels

En el siguiente ejemplo de código, se muestra cómo usar DetectModerationLabels.

Para obtener información, consulte [Detección de imágenes inapropiadas](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DetectModerationLabels](#) la Referencia AWS SDK for Java 2.x de la API.

DetectText

En el siguiente ejemplo de código, se muestra cómo usar DetectText.

Para obtener información, consulte [Detección de texto en una imagen](#).

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DetectText](#) la Referencia AWS SDK for Java 2.x de la API.

IndexFaces

En el siguiente ejemplo de código, se muestra cómo usar IndexFaces.

Para obtener información, consulte [Adición de rostros a una colección](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [IndexFaces](#) la Referencia AWS SDK for Java 2.x de la API.

ListCollections

En el siguiente ejemplo de código, se muestra cómo usar `ListCollections`.

Para obtener información, consulte [Enumerar colecciones](#).

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListCollections](#) la Referencia AWS SDK for Java 2.x de la API.

ListFaces

En el siguiente ejemplo de código, se muestra cómo usar ListFaces.

Para obtener información, consulte [Enumerar rostros en una colección](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListFaces](#) la Referencia AWS SDK for Java 2.x de la API.

RecognizeCelebrities

En el siguiente ejemplo de código, se muestra cómo usar `RecognizeCelebrities`.

Para obtener información, consulte [Reconocimiento de famosos en una imagen](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [RecognizeCelebrities](#) la Referencia AWS SDK for Java 2.x de la API.

SearchFaces

En el siguiente ejemplo de código, se muestra cómo usar SearchFaces.

Para obtener información, consulte [Búsqueda de un rostro \(ID de rostro\)](#).

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [SearchFaces](#) la Referencia AWS SDK for Java 2.x de la API.

SearchFacesByImage

En el siguiente ejemplo de código, se muestra cómo usar SearchFacesByImage.

Para obtener información, consulte [Búsqueda de un rostro \(imagen\)](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [SearchFacesByImage](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Detectar información en los videos

En el siguiente ejemplo de código, se muestra cómo:

- Iniciar trabajos de Amazon Rekognition para detectar elementos como personas, objetos y texto en los videos.
- Compruebe el estado de los trabajos hasta que se terminan.
- Obtener la lista de elementos detectados por cada trabajo.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtenga información sobre famosos a partir de un vídeo ubicado en un bucket de 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(vidOb)
        .build();

    StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
        .startCelebrityRecognition(recognitionRequest);
    startJobId = startCelebrityRecognitionResult.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getCelebrityDetectionResults(RekognitionClient rekClient) {

    try {
        String paginationToken = null;
        GetCelebrityRecognitionResponse recognitionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (recognitionResponse != null)
                paginationToken = recognitionResponse.nextToken();

            GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {
                recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                status = recognitionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
            }
        }
    }
}
```

```

        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetaData = recognitionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<CelebrityRecognition> celebs =
recognitionResponse.celebrities();
    for (CelebrityRecognition celeb : celebs) {
        long seconds = celeb.timestamp() / 1000;
        System.out.print("Sec: " + seconds + " ");
        CelebrityDetail details = celeb.celebrity();
        System.out.println("Name: " + details.name());
        System.out.println("Id: " + details.id());
        System.out.println();
    }

    } while (recognitionResponse.nextToken() != null);

} catch (RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}

```

Detecte etiquetas en un vídeo mediante una operación de detección de etiquetas.

```

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);
        }
    }
}

```

Detecte rostros en un vídeo almacenado en un bucket de 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);
}
}
}

```

Detecte contenido inapropiado u ofensivo en un vídeo almacenado en un bucket de 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 vidObj = Video.builder()
                .s3Object(s3Obj)
                .build();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                .jobTag("Moderation")
                .notificationChannel(channel)
                .video(vidObj)
                .build();

        StartContentModerationResponse startModDetectionResult = rekClient
                .startContentModeration(modDetectionRequest);
        startJobId = startModDetectionResult.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

public static void getModResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetContentModerationResponse modDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (modDetectionResponse != null)
                paginationToken = modDetectionResponse.nextToken();

            GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                modDetectionResponse =
rekClient.getContentModeration(modRequest);
                status = modDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData = modDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
        }
    }
}
```

```

        System.out.println("Job");

        List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
        for (ContentModerationDetection mod : mods) {
            long seconds = mod.timestamp() / 1000;
            System.out.print("Mod label: " + seconds + " ");
            System.out.println(mod.moderationLabel().toString());
            System.out.println();
        }

        } while (modDetectionResponse != null &&
modDetectionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

Detecte segmentos de señales técnicas y segmentos de detección de tomas en un vídeo almacenado en un bucket de 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(vidOb)
        .filters(filters)
        .build();

        StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
        startJobId = segDetectionResponse.jobId();

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getSegmentResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetSegmentDetectionResponse segDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (segDetectionResponse != null)
                paginationToken = segDetectionResponse.nextToken();

            GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();
```

```
// Wait until the job succeeds.
while (!finished) {
    segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
    status = segDetectionResponse.jobStatusAsString();

    if (status.compareTo("SUCCEEDED") == 0)
        finished = true;
    else {
        System.out.println(yy + " status is: " + status);
        Thread.sleep(1000);
    }
    yy++;
}
finished = false;

// Proceed when the job is done - otherwise VideoMetadata is null.
List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
for (VideoMetadata metaData : videoMetaData) {
    System.out.println("Format: " + metaData.format());
    System.out.println("Codec: " + metaData.codec());
    System.out.println("Duration: " + metaData.durationMillis());
    System.out.println("FrameRate: " + metaData.frameRate());
    System.out.println("Job");
}

List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
for (SegmentDetection detectedSegment : detectedSegments) {
    String type = detectedSegment.type().toString();
    if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
        System.out.println("Technical Cue");
        TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
        System.out.println("\tType: " + segmentCue.type());
        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
    }

    if (type.contains(SegmentType.SHOT.toString())) {
        System.out.println("Shot");
        ShotSegment segmentShot = detectedSegment.shotSegment();
```

```

                System.out.println("\tIndex " + segmentShot.index());
                System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
            }

            long seconds = detectedSegment.durationMillis();
            System.out.println("\tDuration : " + seconds + " milliseconds");
            System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
            System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
            System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
            System.out.println();
        }

        } while (segDetectionResponse != null &&
segDetectionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

Detecte texto en un vídeo almacenado en un bucket de 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(RecognitionClient rekClient,
        NotificationChannel channel,
        String bucket,
        String video) {
        try {
            S3Object s3obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidObj = Video.builder()
                .s3Object(s3obj)
                .build();

            StartTextDetectionRequest labelDetectionRequest =
                StartTextDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidObj)
                    .build();

            StartTextDetectionResponse labelDetectionResponse =
                rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RecognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getTextResults(RecognitionClient 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 personas en un vídeo almacenado en un bucket de 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(RecognitionClient rekClient,
        NotificationChannel channel,
        String bucket,
        String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidObj = Video.builder()
                .s3Object(s3Obj)
                .build();

            StartPersonTrackingRequest personTrackingRequest =
                StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidObj)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
                rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RecognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPersonDetectionResults(RecognitionClient 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [GetCelebrityRecognition](#)
 - [GetContentModeration](#)
 - [GetLabelDetection](#)
 - [GetPersonTracking](#)
 - [GetSegmentDetection](#)
 - [GetTextDetection](#)
 - [StartCelebrityRecognition](#)
 - [StartContentModeration](#)
 - [StartLabelDetection](#)
 - [StartPersonTracking](#)
 - [StartSegmentDetection](#)
 - [StartTextDetection](#)

Ejemplos de registro de dominios de Route 53 usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante el AWS SDK for Java 2.x registro de dominios de Route 53.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver ~~las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.~~

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Registro de dominio de Introducción a Route 53

En los siguientes ejemplos de código se muestra cómo empezar a utilizar el registro de dominio de Route 53.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListPrices](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CheckDomainAvailability

En el siguiente ejemplo de código, se muestra cómo usar CheckDomainAvailability.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainAvailabilityResponse response = route53DomainsClient
            .checkDomainAvailability(availabilityRequest);
        System.out.println(domainSuggestion + " is " +
response.availability().toString());


    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [CheckDomainAvailability](#) la Referencia AWS SDK for Java 2.x de la API.

CheckDomainTransferability

En el siguiente ejemplo de código, se muestra cómo usar CheckDomainTransferability.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
            .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
response.transferability().transferable().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [CheckDomainTransferability](#) la Referencia AWS SDK for Java 2.x de la API.

GetDomainDetail

En el siguiente ejemplo de código, se muestra cómo usar GetDomainDetail.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
    try {
        GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
            .domainName(domainSuggestion)
            .build();

        GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
        System.out.println("The contact first name is " +
response.registrantContact().firstName());
        System.out.println("The contact last name is " +
response.registrantContact().lastName());
        System.out.println("The contact org name is " +
response.registrantContact().organizationName());


    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [GetDomainDetail](#) la Referencia AWS SDK for Java 2.x de la API.

GetDomainSuggestions

En el siguiente ejemplo de código, se muestra cómo usar `GetDomainSuggestions`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetDomainSuggestions](#) la Referencia AWS SDK for Java 2.x de la API.

GetOperationDetail

En el siguiente ejemplo de código, se muestra cómo usar `GetOperationDetail`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
    try {
        GetOperationDetailRequest detailRequest =
        GetOperationDetailRequest.builder()
            .operationId(operationId)
            .build();

        GetOperationDetailResponse response =
        route53DomainsClient.getOperationDetail(detailRequest);
        System.out.println("Operation detail message is " + response.message());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [GetOperationDetail](#) la Referencia AWS SDK for Java 2.x de la API.

ListDomains

En el siguiente ejemplo de código, se muestra cómo usar ListDomains.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
            .flatMap(r -> r.domains().stream())
            .forEach(content -> System.out.println("The domain name is " +
content.domainName()));
    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [ListDomains](#) la Referencia AWS SDK for Java 2.x de la API.

ListOperations

En el siguiente ejemplo de código, se muestra cómo usar ListOperations.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);
    }
}
```

```
ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
    .submittedSince(myTime)
    .build();

ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
listRes.stream()
    .flatMap(r -> r.operations().stream())
    .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
        " Status: " + content.statusAsString() +
        " Date: " + content.submittedDate()));

} catch (Route53Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Para obtener más información sobre la API, consulta [ListOperations](#) la Referencia AWS SDK for Java 2.x de la API.

ListPrices

En el siguiente ejemplo de código, se muestra cómo usar `ListPrices`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .tld(domainType)
```

```
        .build());

        ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
        listRes.stream()
            .flatMap(r -> r.prices().stream())
            .forEach(content -> System.out.println(" Name: " +
content.name() +
                " Registration: " + content.registrationPrice().price()
+ " "
                + content.registrationPrice().currency() +
                " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [ListPrices](#) la Referencia AWS SDK for Java 2.x de la API.

RegisterDomain

En el siguiente ejemplo de código, se muestra cómo usar RegisterDomain.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
        String domainSuggestion,
        String phoneNumber,
        String email,
```



```
        String firstName,
        String lastName,
        String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
            .organizationName("My Org")
            .addressLine1("My Address")
            .zipCode("123 123")
            .build();

        RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
            .adminContact(contactDetail)
            .registrantContact(contactDetail)
            .techContact(contactDetail)
            .domainName(domainSuggestion)
            .autoRenew(true)
            .durationInYears(1)
            .build();

        RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
        System.out.println("Registration requested. Operation Id: " +
response.operationId());
        return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Para obtener más información sobre la API, consulta [RegisterDomain](#) la Referencia AWS SDK for Java 2.x de la API.

ViewBilling

En el siguiente ejemplo de código, se muestra cómo usar ViewBilling.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        LocalDateTime localDateTime2 = localDateTime.minusYears(1);
        Instant myStartTime = localDateTime2.toInstant(zoneOffset);
        Instant myEndTime = localDateTime.toInstant(zoneOffset);

        ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
            .start(myStartTime)
            .end(myEndTime)
            .build();

        ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
        listRes.stream()
            .flatMap(r -> r.billingRecords().stream())
            .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                " Operation: " + content.operationAsString() +
                " Price: " + content.price()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
    }
}
```

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

- Para obtener más información sobre la API, consulta [ViewBilling](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Introducción a los dominios

En el siguiente ejemplo de código, se muestra cómo:

- Enumere los dominios actuales y las operaciones del año pasado.
- Consulte la facturación del año pasado y los precios de los tipos de dominio.
- Obtención de sugerencias de dominios.
- Compruebe la disponibilidad y la transferibilidad del dominio.
- Si lo desea, solicite el registro de un dominio.
- Obtención de información de una operación.
- Si lo desea, obtenga información del dominio.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CheckDomainAvailability](#)
 - [CheckDomainTransferability](#)
 - [GetDomainDetail](#)
 - [GetDomainSuggestions](#)
 - [GetOperationDetail](#)
 - [ListDomains](#)
 - [ListOperations](#)
 - [ListPrices](#)
 - [RegisterDomain](#)
 - [ViewBilling](#)

Ejemplos de Amazon S3 usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes AWS SDK for Java 2.x mediante Amazon S3.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Amazon S3

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon S3.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListBuckets](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)
- [Ejemplos sin servidor](#)

Acciones

CopyObject

En el siguiente ejemplo de código, se muestra cómo usar CopyObject.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Copie un objeto con un [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 "";
}
}
}

```

Usa un [S3 TransferManager](#) para [copiar un objeto](#) de un depósito a otro. Vea el [archivo completo](#) y [pruébelo](#).

```

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 obtener más información sobre la API, consulte [CopyObject](#) la Referencia AWS SDK for Java 2.x de la API.

CreateBucket

En el siguiente ejemplo de código, se muestra cómo usar CreateBucket.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear un 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.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);
    }
}
}
```

Cree un nuevo bucket con el bloqueo de objetos habilitado.

```
// Create a new Amazon S3 bucket with object lock options.
```

```
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}
```

- Para obtener más información sobre la API, consulta [CreateBucket](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteBucket

En el siguiente ejemplo de código, se muestra cómo usar DeleteBucket.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

- Para obtener más información sobre la API, consulta [DeleteBucket](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteBucketPolicy

En el siguiente ejemplo de código, se muestra cómo usar DeleteBucketPolicy.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteBucketPolicy](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteBucketWebsite

En el siguiente ejemplo de código, se muestra cómo usar DeleteBucketWebsite.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteBucketWebsite](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteObjects

En el siguiente ejemplo de código, se muestra cómo usar DeleteObjects.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


```
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 putObj;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putObj = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        s3.putObject(putObj, 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 obtener más información sobre la API, consulta [DeleteObjects](#) la Referencia AWS SDK for Java 2.x de la API.

GetBucketAcl

En el siguiente ejemplo de código, se muestra cómo usar `GetBucketAcl`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetBucketAcl](#) la Referencia AWS SDK for Java 2.x de la API.

GetBucketPolicy

En el siguiente ejemplo de código, se muestra cómo usar `GetBucketPolicy`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetBucketPolicy](#) la Referencia AWS SDK for Java 2.x de la API.

GetObject

En el siguiente ejemplo de código, se muestra cómo usar `GetObject`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Lea datos como una matriz de bytes con un [S3 Client](#).

```
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);
    }
}
}
}

```

Utilice un [S3 TransferManager](#) para [descargar un objeto](#) de un bucket de S3 a un archivo local. Vea el [archivo completo](#) y [pruébelo](#).

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.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 downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFileWithPath))
            .build();
    }

```

```

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }

```

Lea las etiquetas que pertenecen a un objeto con un [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.getTagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.getKey());
            System.out.println(tag.getValue());
        }

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

Obtenga una URL para un objeto con un [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);
    }
}
}
}

```

Obtenga un objeto mediante el objeto de cliente S3Presigner con un [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.addRequestProperty(header, value);
        });
    });

    // Send any request payload that the service needs (not needed when
    // isBrowserExecutable is true).
    if (presignedGetObjectRequest.signedPayload().isPresent()) {
        connection.setDoOutput(true);

        try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
            OutputStream httpOutputStream =
connection.getOutputStream()) {
            IoUtils.copy(signedPayload, httpOutputStream);
        }
    }

    // Download the result of executing the request.
    try (InputStream content = connection.getInputStream()) {
        System.out.println("Service returned response: ");
        IoUtils.copy(content, System.out);
    }

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
}
}

```

Obtenga un objeto mediante un `ResponseTransformer` objeto y [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 obtener más información sobre la API, consulte la Referencia de [GetObject](#) la AWS SDK for Java 2.x API.

GetObjectLegalHold

En el siguiente ejemplo de código, se muestra cómo usar `GetObjectLegalHold`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
        "'");
    }

    return null;
}
```

- Para obtener más información sobre la API, consulta [GetObjectLegalHold](#) la Referencia AWS SDK for Java 2.x de la API.

GetObjectLockConfiguration

En el siguiente ejemplo de código, se muestra cómo usar `GetObjectLockConfiguration`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .build();

    GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName+": ");
    System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().getObjectLockEnabled());
    System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}
```

- Para obtener más información sobre la API, consulta [GetObjectLockConfiguration](#) la Referencia AWS SDK for Java 2.x de la API.

GetObjectRetention

En el siguiente ejemplo de código, se muestra cómo usar `GetObjectRetention`.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();
```

```

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("tObject retention for "+key +" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+ ".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

```

- Para obtener más información sobre la API, consulta [GetObjectRetention](#) la Referencia AWS SDK for Java 2.x de la API.

HeadObject

En el siguiente ejemplo de código, se muestra cómo usar HeadObject.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Determinar el tipo de contenido de un 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);
    }
}
}
```

Obtener el estado de restauración de un 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 obtener más información sobre la API, consulta [HeadObject](#) la Referencia AWS SDK for Java 2.x de la API.

ListBuckets

En el siguiente ejemplo de código, se muestra cómo usar ListBuckets.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListBuckets](#) la Referencia AWS SDK for Java 2.x de la API.

ListMultipartUploads

En el siguiente ejemplo de código, se muestra cómo usar ListMultipartUploads.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


```
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 obtener más información sobre la API, consulta [ListMultipartUploads](#) la Referencia AWS SDK for Java 2.x de la API.

ListObjectsV2

En el siguiente ejemplo de código, se muestra cómo usar ListObjectsV2.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 " + calKb(myValue.size()) + "
KBs");

        System.out.print("\n The owner is " + myValue.owner());
    }

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

// convert bytes to kbs.
private static long calKb(Long val) {
    return val / 1024;
}
}

```

Muestre objetos mediante paginación.

```

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 obtener más información sobre la API, consulta la [ListObjectsversión 2](#) en la referencia de la AWS SDK for Java 2.x API.

PutBucketAc1

En el siguiente ejemplo de código, se muestra cómo usar PutBucketAc1.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

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 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 obtener más información sobre la API, consulta [PutBucketCors](#) en la Referencia AWS SDK for Java 2.x de la API.

PutBucketCors

En el siguiente ejemplo de código, se muestra cómo usar PutBucketCors.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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.getCORSRules();  
      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 obtener más información sobre la API, consulta [PutBucketCors](#) la Referencia AWS SDK for Java 2.x de la API.

PutBucketLifecycleConfiguration

En el siguiente ejemplo de código, se muestra cómo usar `PutBucketLifecycleConfiguration`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [PutBucketLifecycleConfiguration](#) la Referencia AWS SDK for Java 2.x de la API.

PutBucketNotificationConfiguration

En el siguiente ejemplo de código, se muestra cómo usar `PutBucketNotificationConfiguration`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [PutBucketNotificationConfiguration](#) la Referencia AWS SDK for Java 2.x de la API.

PutBucketPolicy

En el siguiente ejemplo de código, se muestra cómo usar PutBucketPolicy.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [PutBucketPolicy](#) la Referencia AWS SDK for Java 2.x de la API.

PutBucketWebsite

En el siguiente ejemplo de código, se muestra cómo usar PutBucketWebsite.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [PutBucketWebsite](#) la Referencia AWS SDK for Java 2.x de la API.

PutObject

En el siguiente ejemplo de código, se muestra cómo usar PutObject.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cargue un archivo en un bucket con un [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);
        }
    }
}
```

Usa un [S3 TransferManager](#) para [subir un archivo](#) a un bucket. Vea el [archivo completo](#) y [pruébelo](#).

```
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))
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

Cargue un objeto en un bucket y configure las etiquetas con un [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;
}
}
```

Cargue un objeto en un bucket y configure los metadatos con un [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);
    }
}

```

Cargue un objeto en un bucket y configure un valor de retención de objetos con un [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 obtener más información sobre la API, consulte [PutObject](#) la Referencia AWS SDK for Java 2.x de la API.

PutObjectLegalHold

En el siguiente ejemplo de código, se muestra cómo usar PutObjectLegalHold.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()

```

```
        .status(ObjectLockLegalHoldStatus.ON)
        .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+ ".");
    }
```

- Para obtener más información sobre la API, consulta [PutObjectLegalHold](#) la Referencia AWS SDK for Java 2.x de la API.

PutObjectLockConfiguration

En el siguiente ejemplo de código, se muestra cómo usar PutObjectLockConfiguration.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Establecimiento de la configuración de bloqueo de objetos de un bucket

```
// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
```

```
VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
    .status(BucketVersioningStatus.ENABLED)
    .build();

PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
    .bucket(bucketName)
    .versioningConfiguration(versioningConfiguration)
    .build();

// Enable versioning on the bucket.
getClient().putBucketVersioning(putBucketVersioningRequest);
PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .build())
    .build();

getClient().putObjectLockConfiguration(request);
System.out.println("Successfully enabled object lock on "+bucketName);

} catch (S3Exception ex) {
    System.out.println("Error modifying object lock: '" + ex.getMessage() +
    "");
}
}
```

Establecimiento del período de retención predeterminado de un bucket

```
// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
    .mfaDelete(MFADelete.DISABLED)
    .status(BucketVersioningStatus.ENABLED)
    .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
```

```
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention retention = DefaultRetention.builder()
        .days(1)
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(retention)
        .build();

    ObjectLockConfiguration objectLockConfiguration =
    ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .rule(lockRule)
        .build();

    PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
    PutObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .objectLockConfiguration(objectLockConfiguration)
        .build();


    getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
    System.out.println("Added a default retention to bucket "+bucketName +".");
}
```

- Para obtener más información sobre la API, consulta [PutObjectLockConfiguration](#) la Referencia AWS SDK for Java 2.x de la API.

PutObjectRetention

En el siguiente ejemplo de código, se muestra cómo usar PutObjectRetention.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time zone.
    ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);

    // Print the formatted date string.
    System.out.println("Formatted Date: " + humanReadableDate);
    ObjectLockRetention retention = ObjectLockRetention.builder()
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .retainUntilDate(futureInstant)
        .build();

    PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .retention(retention)
    .build();

    getClient().putObjectRetention(retentionRequest);
    System.out.println("Set retention for "+objectKey+" in "+bucketName+"
until "+humanReadableDate+".");
}
```

- Para obtener más información sobre la API, consulta [PutObjectRetention](#) la Referencia AWS SDK for Java 2.x de la API.

RestoreObject

En el siguiente ejemplo de código, se muestra cómo usar RestoreObject.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [RestoreObject](#) la Referencia AWS SDK for Java 2.x de la API.

SelectObjectContent

En el siguiente ejemplo de código, se muestra cómo usar `SelectObjectContent`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

En el siguiente ejemplo, se muestra una consulta utilizando un objeto JSON. En el [ejemplo completo](#), también se muestra el uso de un 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 obtener más información sobre la API, consulta [SelectObjectContent](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Crear una URL prefirmada

En el siguiente ejemplo de código se muestra cómo crear una URL prefirmada para Amazon S3 y cargar un objeto.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Genere una URL prefirmada para un objeto y, a continuación, descárguela (solicitud GET).

Importaciones.

```
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;
```

Genere la 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();
    }
}
```

Descargue el objeto mediante uno de los tres enfoques siguientes.

Utilice la clase JDK `HttpURLConnection` (desde la versión 1.1) para realizar la descarga.

```
/* 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();
    }

```

Utilice la clase JDK `HttpClient` (desde la versión 11) para realizar la descarga.

```

/* 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();
}

```

Usa la `SdkHttpClient` clase AWS SDK for Java para realizar la descarga.

```

/* 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();  
}
```

Genere una URL prefirmada para una carga y, a continuación, cargue un archivo (solicitud PUT).

Importaciones.

```
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;
```

Genere la 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();
    }
}
```

Cargue un objeto de archivo mediante uno de los tres enfoques siguientes.

Utilice la clase `JDK HttpURLConnection` (desde la versión 1.1) para realizar la carga.

```
/* 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);
    }
}

```

Utilice la clase JDK `HttpClient` (desde la versión 11) para realizar la carga.

```

/* 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);
}
}

```

Usa la `SdkHttpClient` clase AWS for Java V2 para realizar la carga.

```

/* 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);
    }
}
```

Eliminación de cargas multiparte incompletas

En el siguiente ejemplo de código, se muestra cómo eliminar o detener cargas multiparte incompletas de Amazon S3.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Para detener las cargas multiparte que estén en curso o incompletas por cualquier motivo, puede obtener una lista de las cargas y, a continuación, eliminarlas, tal y como se muestra en el siguiente ejemplo.

```
public static void abortIncompleteMultipartUploadsFromList() {
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
```

```

        .bucket(bucketName)
        .key(upload.key())
        .expectedBucketOwner(accountId)
        .uploadId(upload.uploadId())
        .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
upload.uploadId(), bucketName);
        }
    }
}

```

Para eliminar las cargas multiparte incompletas que se hayan iniciado antes o después de una fecha, puede eliminar de forma selectiva las cargas multiparte en función de un momento dado, como se muestra en el siguiente ejemplo.

```

static void abortIncompleteMultipartUploadsOlderThan(Instant pointInTime) {
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        logger.info("Found multipartUpload with upload ID [{}], initiated [{}]",
upload.uploadId(), upload.initiated());
        if (upload.initiated().isBefore(pointInTime)) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

```

```

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}
}
}

```

Si tiene acceso al ID de carga después de iniciar una carga multiparte, puede eliminar la carga en curso utilizando ese ID.

```

static void abortMultipartUploadUsingUploadId() {
    String uploadId = startUploadReturningUploadId();
    AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -> b
        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
    }
}
}

```

Para eliminar de forma sistemática las cargas multiparte incompletas que tengan más de un número determinado de días, establezca una configuración de ciclo de vida de bucket para el bucket. En el siguiente ejemplo, se muestra cómo crear una regla para eliminar cargas incompletas que tienen más de 7 días.

```

static void abortMultipartUploadsUsingLifecycleConfig() {
    Collection<LifecycleRule> lifeCycleRules = List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());
}

```

```
// If the action is successful, the service sends back an HTTP 200 response
with an empty HTTP body.
    PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
    } else {
        logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
    }
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [PutBucketLifecycleConfiguration](#)

Descargar objetos en un directorio local

El siguiente ejemplo de código muestra cómo descargar todos los objetos de un bucket de Amazon Simple Storage Service (Amazon S3) en un directorio local.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Utilice un [S3 TransferManager](#) para [descargar todos los objetos de S3](#) en el mismo depósito de S3. Vea el [archivo completo](#) y [pruébelo](#).

```
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 obtener más información sobre la API, consulte [DownloadDirectory](#) la Referencia AWS SDK for Java 2.x de la API.

Comenzar a usar buckets y objetos

En el siguiente ejemplo de código, se muestra cómo:

- Creación de un bucket y cargar un archivo en el bucket.

- Descargar un objeto desde un bucket.
- Copiar un objeto en una subcarpeta de un bucket.
- Obtención de una lista de los objetos de un bucket.
- Eliminación del bucket y todos los objetos que incluye.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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. Update a local file to the Amazon S3 bucket.");  
uploadLocalFile(s3, bucketName, key, objectPath);  
System.out.println(DASHES);
```

```
System.out.println(DASHES);  
System.out.println("3. Download the object to another local file.");
```

```
    getObjectBytes(s3, bucketName, key, savePath);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Perform a multipart upload.");
    String multipartKey = "multiPartKey";
    multipartUpload(s3, toBucket, multipartKey);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. List all objects located in the Amazon S3 bucket.");
    listAllObjects(s3, bucketName);
    anotherListExample(s3, bucketName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Copy the object to another Amazon S3 bucket.");
    copyBucketObject(s3, bucketName, key, toBucket);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Delete the object from the Amazon S3 bucket.");
    deleteObjectFromBucket(s3, bucketName, key);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("8. Delete the Amazon S3 bucket.");
    deleteBucket(s3, bucketName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("All Amazon S3 operations were successfully performed");
    System.out.println(DASHES);
    s3.close();
}

// Create a bucket by using a S3Waiter object.
public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();
    }
}
```

```
s3Client.createBucket(bucketRequest);
HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
    .bucket(bucketName)
    .build();

// Wait until the bucket is created and print out the response.
WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println(bucketName + " is ready");

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

public static void deleteBucket(S3Client client, String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    client.deleteBucket(deleteBucketRequest);
    System.out.println(bucket + " was deleted.");
}

/**
 * Upload an object in parts.
 */
public static void multipartUpload(S3Client s3, String bucketName, String key) {
    int mB = 1024 * 1024;
    // First create a multipart upload and get the upload id.
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

    CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
    String uploadId = response.uploadId();
    System.out.println(uploadId);

    // Upload all the different parts of the object.
```

```
UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .partNumber(1).build();

String etag1 = s3.uploadPart(uploadPartRequest1,
    RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
    .eTag();
CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
    .uploadId(uploadId)
    .partNumber(2).build();
String etag2 = s3.uploadPart(uploadPartRequest2,
    RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
    .eTag();
CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

// Call completeMultipartUpload operation to tell S3 to merge all uploaded
// parts and finish the multipart operation.
CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
    .parts(part1, part2)
    .build();

CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

s3.completeMultipartUpload(completeMultipartUploadRequest);
}

private static ByteBuffer getRandomByteBuffer(int size) {
    byte[] b = new byte[size];
    new Random().nextBytes(b);
    return ByteBuffer.wrap(b);
}
```

```
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

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

    public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
    }

    public static void listAllObjects(S3Client s3, String bucketName) {
        ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
```

```
        .build();

    boolean done = false;
    while (!done) {
        ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
        for (S3Object content : listObjResponse.contents()) {
            System.out.println(content.key());
        }

        if (listObjResponse.nextContinuationToken() == null) {
            done = true;
        }

        listObjectsReqManual = listObjectsReqManual.toBuilder()
            .continuationToken(listObjResponse.nextContinuationToken())
            .build();
    }
}

public static void anotherListExample(S3Client s3, String bucketName) {
    ListObjectsV2Request listReq = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

    // Process response pages.
    listRes.stream()
        .flatMap(r -> r.contents().stream())
        .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

    // Helper method to work with paginated collection of items directly.
    listRes.contents().stream()
        .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

    for (S3Object content : listRes.contents()) {
        System.out.println(" Key: " + content.key() + " size = " +
content.size());
    }
}
```



```
public static void deleteObjectFromBucket(S3Client s3, String bucketName, String
key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.deleteObject(deleteObjectRequest);
    System.out.println(key + " was deleted");
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    String encodedUrl = null;
    try {
        encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());
    } catch (UnsupportedEncodingException e) {
        System.out.println("URL could not be encoded: " + e.getMessage());
    }
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .copySource(encodedUrl)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        System.out.println("The " + objectKey + " was copied to " + toBucket);
        return copyRes.copyObjectResult().toString();

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


- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Obtención de la configuración de retención legal de un objeto

En el siguiente ejemplo de código, se muestra cómo obtener la configuración de retención legal de un bucket de S3.

SDK para Java 2.x

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();
    } catch (S3Exception ex) {
```

```
        System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
        """);
    }

    return null;
}
```

- Para obtener más información sobre la API, consulta [GetObjectLegalHold](#) la Referencia AWS SDK for Java 2.x de la API.

Bloqueo de objetos de Amazon S3

En el siguiente ejemplo de código, se muestra cómo trabajar con características de bloqueo de objetos de S3.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute un escenario interactivo en el que se demuestren las características de bloqueo de objetos de Amazon S3.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
Before running this Java V2 code example, set up your development
environment, including your credentials.
```

For more information, see the following documentation topic:

<https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html>

This Java example performs the following tasks:

1. Create test Amazon Simple Storage Service (S3) buckets with different lock policies.
2. Upload sample objects to each bucket.
3. Set some Legal Hold and Retention Periods on objects and buckets.
4. Investigate lock policies by viewing settings or attempting to delete or overwrite objects.
5. Clean up objects and buckets.

```
*/
```

```
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        // Get the current date and time to ensure bucket name is unique.
        LocalDateTime currentTime = LocalDateTime.now();

        // Format the date and time as a string.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
        String timeStamp = currentTime.format(formatter);

        s3LockActions = new S3LockActions();
        bucketName = "bucket"+timeStamp;
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3) Object
Locking Workflow Scenario.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
    }
}
```

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

System.out.println(DASHES);
System.out.println("Lets present the user with choices.");
System.out.println("Press Enter to continue...");
scanner.nextLine();
demoActionChoices() ;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Would you like to clean up the resources? (y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    cleanup();
    System.out.println("Clean up is complete.");
}

System.out.println("Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Amazon S3 Object Locking Workflow is complete.");
System.out.println(DASHES);
}

// Present the user with the demo action choices.
public static void demoActionChoices() {
    String[] choices = {
        "List all files in buckets.",
        "Attempt to delete a file.",
        "Attempt to delete a file with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the object and bucket retention settings for a file.",
        "View the legal hold settings for a file.",
        "Finish the workflow."
    };

    int choice = 0;
    while (true) {
        System.out.println(DASHES);
        choice = getChoiceResponse("Explore the S3 locking features by selecting
one of the following choices:", choices);
    }
}
```

```
System.out.println(DASHES);
System.out.println("You selected "+choices[choice]);
switch (choice) {
    case 0 -> {
        s3LockActions.listBucketsAndObjects(bucketNames, true);
    }

    case 1 -> {
        System.out.println("Enter the number of the object to delete:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        String version = allFiles.get(fileChoice).getVersion();
        s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
    }

    case 2 -> {
        System.out.println("Enter the number of the object to delete:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        String version = allFiles.get(fileChoice).getVersion();
        s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
    }

    case 3 -> {
        System.out.println("Enter the number of the object to
overwrite:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
```

```
String[] fileKeysArray = fileKeys.toArray(new String[0]);
int fileChoice = getChoiceResponse(null, fileKeysArray);
String objectKey = fileKeys.get(fileChoice);
String bucketName = allFiles.get(fileChoice).getBucketName();

// Attempt to overwrite the file.
try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
    writer.write("This is a modified text.");

} catch (IOException e) {
    e.printStackTrace();
}
s3LockActions.uploadFile(bucketName, objectKey, objectKey);
}

case 4 -> {
    System.out.println("Enter the number of the object to
overwrite:");

    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
    String[] fileKeysArray = fileKeys.toArray(new String[0]);
    int fileChoice = getChoiceResponse(null, fileKeysArray);
    String objectKey = fileKeys.get(fileChoice);
    String bucketName = allFiles.get(fileChoice).getBucketName();
    s3LockActions.getObjectRetention(bucketName, objectKey);
}

case 5 -> {
    System.out.println("Enter the number of the object to view:");
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
    String[] fileKeysArray = fileKeys.toArray(new String[0]);
    int fileChoice = getChoiceResponse(null, fileKeysArray);
    String objectKey = fileKeys.get(fileChoice);
    String bucketName = allFiles.get(fileChoice).getBucketName();
    s3LockActions.getObjectLegalHold(bucketName, objectKey);
    s3LockActions.getBucketObjectLockConfiguration(bucketName);
}
```

```

        case 6 -> {
            System.out.println("Exiting the workflow...");
            return;
        }

        default -> {
            System.out.println("Invalid choice. Please select again.");
        }
    }
}

// Clean up the resources from the scenario.
private static void cleanup() {
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
    for (S3InfoObject fileInfo : allFiles) {
        String bucketName = fileInfo.getBucketName();
        String key = fileInfo.getKeyName();
        String version = fileInfo.getVersion();
        if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
            ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
            if (legalHold != null) {
                String holdStatus = legalHold.status().name();
                System.out.println(holdStatus);
                if (holdStatus.compareTo("ON") == 0) {
                    s3LockActions.modifyObjectLegalHold(bucketName, key, false);
                }
            }
            // Check for a retention period.
            ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
            boolean hasRetentionPeriod ;
            hasRetentionPeriod = retention != null;
            s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

        } else {
            System.out.println(bucketName + " objects do not have a legal lock");
            s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
        }
    }
}

```



```
    }

    // Delete the buckets.
    System.out.println("Delete "+bucketName);
    for (String bucket : bucketNames){
        s3LockActions.deleteBucketByName(bucket);
    }
}

private static void setup() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("""
        For this workflow, we will use the AWS SDK for Java to create
several S3
        buckets and files to demonstrate working with S3 locking features.
        """);

    System.out.println("S3 buckets can be created either with or without object
lock enabled.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();

    // Create three S3 buckets.
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
    s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Bucket "+bucketNames.get(2) +" will be configured to use
object locking with a default retention period.");
    s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
    s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    // Upload some files to the buckets.
    System.out.println("Now let's add some test files:");
    String fileName = "exampleFile.txt";
```

```
int fileCount = 2;
try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
    writer.write("This is a sample file for uploading to a bucket.");

} catch (IOException e) {
    e.printStackTrace();
}

for (String bucketName : bucketNames){
    for (int i = 0; i < fileCount; i++) {
        // Get the file name without extension.
        String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
        int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
        if (extensionIndex > 0) {
            fileNameWithoutExtension = fileNameWithoutExtension.substring(0,
extensionIndex);
        }

        // Create the numbered file names.
        String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
        fileNames.add(numberedFileName);
        s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
    }
}

String question = null;
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println("Now we can set some object lock policies on individual
files:");
for (String bucketName : bucketNames) {
    for (int i = 0; i < fileNames.size(); i++){

        // No modifications to the objects in the first bucket.
        if (!bucketName.equals(bucketNames.get(0))) {
            String exampleFileName = fileNames.get(i);
            switch (i) {
                case 0 -> {
                    question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                    System.out.println(question);
```

```

        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            System.out.println("**** You have selected to put a
legal hold " + exampleFileName);

            // Set a legal hold.
            s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
        }
    }
    case 1 -> {
        ""
        Would you like to add a 1 day Governance retention
period to %s in %s (y/n)?
        Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
        "".formatted(exampleFileName, bucketName);
        System.out.println(question);
        String ans2 = scanner.nextLine().trim();
        if (ans2.equalsIgnoreCase("y")) {

s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
        }
    }
}
}
}
}
}

// Get file extension.
private static String getFileExtension(String fileName) {
    int dotIndex = fileName.lastIndexOf('.');
    if (dotIndex > 0) {
        return fileName.substring(dotIndex);
    }
    return "";
}

public static void configurationSetup() {
    String noLockBucketName = bucketName + "-no-lock";
    String lockEnabledBucketName = bucketName + "-lock-enabled";

```

```
        String retentionAfterCreationBucketName = bucketName + "-retention-after-creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }

        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

Una clase contenedora para funciones de S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
```

```
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
}
```

```
// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time zone.
    ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);

    // Print the formatted date string.
    System.out.println("Formatted Date: " + humanReadableDate);
    ObjectLockRetention retention = ObjectLockRetention.builder()
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .retainUntilDate(futureInstant)
        .build();

    PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .retention(retention)
    .build();

    getClient().putObjectRetention(retentionRequest);
    System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}

// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .build();
```

```
        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
        ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
        "'");
    }

    return null;
}

// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}

public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
    AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
    return bucketNames.stream()
        .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
        .map(version -> {
            S3InfoObject s3InfoObject = new S3InfoObject();
            s3InfoObject.setBucketName(bucketName);
            s3InfoObject.setVersion(version.versionId());
```

```

        s3InfoObject.setKeyName(version.key());
        return s3InfoObject;
    )))
    .peek(s3InfoObject -> {
        int i = counter.incrementAndGet(); // Increment and get the updated
value.
        if (interactive) {
            System.out.println(i + ": " + s3InfoObject.getKeyName());
            System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
            System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
        }
    })
    .collect(Collectors.toList());
}

public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
    ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
        .bucket(bucketName)
        .build();

    return getClient().listObjectVersions(versionsRequest);
}

// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .mfaDelete(MFADelete.DISABLED)
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention retention = DefaultRetention.builder()
        .days(1)

```



```
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(rention)
        .build();

    ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .rule(lockRule)
        .build();

    PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .objectLockConfiguration(objectLockConfiguration)
        .build();

    getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
    System.out.println("Added a default retention to bucket "+bucketName +".");
}

// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        // Enable versioning on the bucket.
        getClient().putBucketVersioning(putBucketVersioningRequest);
        PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
```

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

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on "+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage() +
        """);
    }
}

public void uploadFile(String bucketName, String objectName, String filePath) {
    Path file = Paths.get(filePath);
    PutObjectRequest request = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectName)
        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
        .build();

    PutObjectResponse response = getClient().putObject(request, file);
    if (response != null) {
        System.out.println("\tSuccessfully uploaded " + objectName + " to " +
        bucketName + ".");
    } else {
        System.out.println("\tCould not upload " + objectName + " to " +
        bucketName + ".");
    }
}

// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }
}
```

```
        PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .legalHold(legalHold)
    .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey, boolean
hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

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

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
```

```
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("tObject retention for "+key+" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+".");

        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

public void deleteBucketByName(String bucketName) {
    try {
        DeleteBucketRequest request = DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        getClient().deleteBucket(request);
        System.out.println(bucketName +" was deleted.");

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

// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .build();

    GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName +": ");
}
```

```
        System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
    }
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Analizar los URI

En el siguiente ejemplo se muestra cómo analizar los URI de Amazon S3 para extraer componentes importantes como el nombre del bucket y la clave de objeto.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Analice un URI de Amazon S3 mediante la clase [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);
    }
}
```

Ejecución de una carga multiparte

En el siguiente ejemplo de código, se muestra cómo realizar una carga multiparte a un objeto de Amazon S3.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

En los ejemplos de código se utilizan las siguientes importaciones.

```
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.S3AsyncClient;
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.PutObjectResponse;
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;
import java.util.concurrent.CompletableFuture;
```

Utilice el [Gestor de transferencias de Amazon S3](#) situado sobre el [cliente S3 basado en CRT de AWS](#) para realizar de forma transparente una carga multiparte cuando el tamaño del contenido supere un umbral. El umbral de tamaño predeterminado es 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();
}
```

Utilice la [API S3Client](#) para realizar una carga multiparte.

```
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();
    long position = 0;
    while (position < fileSize) {
        file.seek(position);
        long 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()));
}
```

Utilice la [AsyncClient API de S3](#) con soporte multipart habilitado para realizar una carga multiparte.

```
public void multipartUploadWithS3AsyncClient(String filePath) {
    // Enable multipart support.
    S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
        .multipartEnabled(true)
        .build();

    CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b ->
b        .bucket(bucketName)
        .key(key),
        Paths.get(filePath));

    response.join();
    logger.info("File uploaded in multiple 8 MiB parts using S3AsyncClient.");
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Realización de un seguimiento de cargas y descargas

En el siguiente ejemplo de código se muestra cómo realizar un seguimiento de cargas o descargas de objetos de Amazon S3.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Realice el seguimiento del progreso de una carga de archivos.

```
public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .source(Paths.get(filePathURI))
        .build();
```

```
FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
```

```
fileUpload.completionFuture().join();
```

```
/*
```

The SDK provides a `LoggingTransferListener` implementation of the `TransferListener` interface.

You can also implement the interface to provide your own logic.

Configure log4J2 with settings such as the following.

```
<Configuration status="WARN">
    <Appenders>
        <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
            <PatternLayout pattern="%m%n"/>
        </Console>
    </Appenders>

    <Loggers>
        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
            <AppenderRef ref="AlignedConsoleAppender"/>
        </logger>
    </Loggers>
</Configuration>
```

Log4J2 logs the progress. The following is example output for a 21.3 MB file upload.

```
Transfer initiated...
|                               | 0.0%
|====                          | 21.1%
|=====                        | 60.5%
|=====                        | 100.0%
```

```

        Transfer complete!
    */
}

```

Realice el seguimiento del progreso de una descarga de archivos.

```

public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                               String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .destination(Paths.get(downloadedFilePath))
        .build();

```

```

    FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

```

```

    CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();

```

```

/*

```

The SDK provides a `LoggingTransferListener` implementation of the `TransferListener` interface.

You can also implement the interface to provide your own logic.

Configure log4J2 with settings such as the following.

```

<Configuration status="WARN">
    <Appenders>
        <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
            <PatternLayout pattern="%m%n"/>
        </Console>
    </Appenders>

    <Loggers>
        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
            <AppenderRef ref="AlignedConsoleAppender"/>
        </logger>
    </Loggers>
</Configuration>

```

Log4J2 logs the progress. The following is example output for a 21.3 MB file download.

```

    Transfer initiated...
    |=====          | 39.4%
    |=====          | 78.8%
    |=====          | 100.0%
    Transfer complete!

    */
}

```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
 - [GetObject](#)
 - [PutObject](#)

Cargar directorio en un bucket

El siguiente ejemplo de código muestra cómo cargar de forma recursiva un directorio local en un bucket de Amazon Simple Storage Service (Amazon S3).

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Usa un [S3 TransferManager](#) para [cargar un directorio local](#). Vea el [archivo completo](#) y [pruébelo](#).

```

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 obtener más información sobre la API, consulte [UploadDirectory](#) la Referencia AWS SDK for Java 2.x de la API.

Cargar o descargar archivos grandes

En el siguiente ejemplo de código se muestra cómo cargar o descargar archivos grandes en y desde Amazon S3.

Para obtener información, consulte [Carga de un objeto con carga multiparte](#).

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Llama a funciones que transfieren archivos desde y hacia un bucket de S3 mediante el S3TransferManager.

```

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();
}

```

Cargar un directorio local completo.

```

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();
}

```

Cargar un solo archivo.

```

public String uploadFile(S3TransferManager transferManager, String bucketName,
    String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))

```



```
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

Carga de un flujo de tamaño desconocido

En los siguientes ejemplos de código, se muestra cómo cargar un flujo de tamaño desconocido en un objeto de Amazon S3.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Utilice el [Cliente S3 basado en CRT de 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 s33CrtAsyncClient,
String bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    CompletableFuture<PutObjectResponse> responseFuture =
        s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
body);

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

    PutObjectResponse response = responseFuture.join(); // Wait for the
response.
    logger.info("Object {} uploaded to bucket {}.", key, bucketName);
    return response;
}
}

```

Utilice el [Gestor de transferencias de 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 sumas de comprobación

En el siguiente ejemplo de código, se muestra cómo utilizar sumas de comprobación para trabajar con un objeto de Amazon S3.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

En los ejemplos de código se utiliza un subconjunto de las siguientes importaciones.

```
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 un algoritmo de suma de comprobación para el método `putObject` al [crear la `PutObjectRequest`](#).

```
public void putObjectWithChecksum() {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```

Compruebe la suma de comprobación del `getObject` método cuando [cree el `GetObjectRequest`](#).

```
public GetObjectResponse getObjectWithChecksum() {
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

Calcule previamente una suma de comprobación para el método `putObject` cuando [cree la `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)));
}
```

```
}

```

Utilice el [Gestor de transferencias de Amazon S3](#) situado sobre el [cliente S3 basado en CRT de AWS](#) para realizar de forma transparente una carga multiparte cuando el tamaño del contenido supere un umbral. El umbral de tamaño predeterminado es 8 MB.

Puede especificar un algoritmo de suma de comprobación para que lo utilice el SDK. De forma predeterminada, el SDK usa el 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();
}

```

Utilice la API [S3Client](#) o (la [AsyncClient API S3](#)) para realizar una carga de varias partes. Si especifica una suma de comprobación adicional, debe especificar el algoritmo que se utilizará al iniciar la carga. También debe especificar el algoritmo para cada solicitud de parte y proporcionar la suma de comprobación calculada para cada parte una vez cargada.

```
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();
    long position = 0;
    while (position < fileSize) {
        file.seek(position);
        long 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Ejemplos sin servidor

Invocación de una función de Lambda desde un desencadenador de Amazon S3

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al cargar un objeto en un bucket de S3. La función recupera el nombre del bucket de S3 y la clave del objeto del parámetro de evento y llama a la API de Amazon S3 para recuperar y registrar el tipo de contenido del objeto.

SDK para Java 2.x

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de S3 con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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);
    }
}
```

Ejemplos de S3 Glacier con SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK for Java 2.x uso de S3 Glacier.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

CreateVault

En el siguiente ejemplo de código, se muestra cómo usar CreateVault.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateVault](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteArchive

En el siguiente ejemplo de código, se muestra cómo usar DeleteArchive.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteArchive](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteVault

En el siguiente ejemplo de código, se muestra cómo usar DeleteVault.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteVault](#) la Referencia AWS SDK for Java 2.x de la API.

InitiateJob

En el siguiente ejemplo de código, se muestra cómo usar `InitiateJob`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Recupera el inventario de una bóveda.

```
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 obtener más información sobre la API, consulte [InitiateJob](#) la referencia AWS SDK for Java 2.x de la API.

ListVaults

En el siguiente ejemplo de código, se muestra cómo usar `ListVaults`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListVaults](#) la Referencia AWS SDK for Java 2.x de la API.

UploadArchive

En el siguiente ejemplo de código, se muestra cómo usar UploadArchive.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [UploadArchive](#) la Referencia AWS SDK for Java 2.x de la API.

SageMaker ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with SageMaker.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

¿Hola SageMaker

Los siguientes ejemplos de código muestran cómo empezar a usarlo SageMaker.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSageMaker {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
            .region(region)
            .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 obtener más información sobre la API, consulta [ListNotebookInstances](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreatePipeline

En el siguiente ejemplo de código, se muestra cómo usar CreatePipeline.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 obtener más información sobre la API, consulta [CreatePipeline](#) la Referencia AWS SDK for Java 2.x de la API.

DeletePipeline

En el siguiente ejemplo de código, se muestra cómo usar DeletePipeline.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
    DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
        .pipelineName(pipelineName)
        .build();

    sageMakerClient.deletePipeline(pipelineRequest);
    System.out.println("*** Successfully deleted " + pipelineName);
}
```

- Para obtener más información sobre la API, consulta [DeletePipeline](#) la Referencia AWS SDK for Java 2.x de la API.

DescribePipelineExecution

En el siguiente ejemplo de código, se muestra cómo usar DescribePipelineExecution.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Check the status of a pipeline execution.
public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
    throws InterruptedException {
    String status;
    int index = 0;
    do {
        DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
            .pipelineExecutionArn(executionArn)
            .build();

        DescribePipelineExecutionResponse response = sageMakerClient
            .describePipelineExecution(pipelineExecutionRequest);
        status = response.pipelineExecutionStatusAsString();
        System.out.println(index + ". The Status of the pipeline is " + status);
        TimeUnit.SECONDS.sleep(4);
        index++;
    } while ("Executing".equals(status));
    System.out.println("Pipeline finished with status " + status);
}
```

- Para obtener más información sobre la API, consulta [DescribePipelineExecution](#) la Referencia AWS SDK for Java 2.x de la API.

StartPipelineExecution

En el siguiente ejemplo de código, se muestra cómo usar `StartPipelineExecution`.

SDK para Java 2.x

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl,
    String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting().create();

    // Set up all parameters required to start the pipeline.
    List<Parameter> parameters = new ArrayList<>();
    Parameter para1 = Parameter.builder()
        .name("parameter_execution_role")
        .value(roleArn)
        .build();

    Parameter para2 = Parameter.builder()
        .name("parameter_queue_url")
        .value(queueUrl)
        .build();

    String inputJSON = "{\n" +
        "  \"DataSourceConfig\": {\n" +
        "    \"S3Data\": {\n" +
        "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
        "    },\n" +
        "    \"Type\": \"S3_DATA\"\n" +
```

```
        " },\n" +
        "  \"DocumentType\": \"CSV\"\n" +
        "}";

System.out.println(inputJSON);

Parameter para3 = Parameter.builder()
    .name("parameter_vej_input_config")
    .value(inputJSON)
    .build();

// Create an ExportVectorEnrichmentJobOutputConfig object.
VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
    .s3Uri(output)
    .build();

ExportVectorEnrichmentJobOutputConfig outputConfig =
ExportVectorEnrichmentJobOutputConfig.builder()
    .s3Data(jobS3Data)
    .build();

String gson4 = gson.toJson(outputConfig);
Parameter para4 = Parameter.builder()
    .name("parameter_vej_export_config")
    .value(gson4)
    .build();
System.out.println("parameter_vej_export_config:" +
gson.toJson(outputConfig));

// Create a VectorEnrichmentJobConfig object.
ReverseGeocodingConfig reverseGeocodingConfig =
ReverseGeocodingConfig.builder()
    .xAttributeName("Longitude")
    .yAttributeName("Latitude")
    .build();

VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
    .reverseGeocodingConfig(reverseGeocodingConfig)
    .build();

String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
Parameter para5 = Parameter.builder()
    .name("parameter_step_1_vej_config")
```



```
        .value(para5JSON)
        .build();

System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
parameters.add(para1);
parameters.add(para2);
parameters.add(para3);
parameters.add(para4);
parameters.add(para5);

StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
return response.pipelineExecutionArn();
}
```

- Para obtener más información sobre la API, consulta [StartPipelineExecution](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Introducción a las tareas y las canalizaciones geoespaciales

En el siguiente ejemplo de código, se muestra cómo:

- Configurar los recursos de una canalización
- Configurar una canalización que ejecuta un trabajo geoespacial
- Iniciar la ejecución de una canalización.
- Supervisar el estado de la ejecución.
- Ver el resultado de la canalización.
- Limpiar recursos.

Para obtener más información, consulte [Crear y ejecutar SageMaker canalizaciones mediante AWS SDK en Community.aws](#).

SDK para Java 2.x

Note

Hay más información sobre. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 to for AWS SageMaker to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
```

```

        .roleName(roleName)
        .build();

        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createSageMakerRole(iam, roleName);
    }
    return roleArn;
}

private static String[] getSageMakerRolePolicies() {
    String[] sageMakerRolePolicies = new String[3];
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
    "AmazonSageMakerGeospatialFullAccess";
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
    return sageMakerRolePolicies;
}

private static String[] getLambdaRolePolicies() {
    String[] lambdaRolePolicies = new String[5];
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
    "AmazonSageMakerGeospatialFullAccess";
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
        + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
    "AWSLambdaSQSQueueExecutionRole";
    return lambdaRolePolicies;
}
}

```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreatePipeline](#)

- [DeletePipeline](#)
- [DescribePipelineExecution](#)
- [StartPipelineExecution](#)
- [UpdatePipeline](#)

Ejemplos de Secrets Manager usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Secrets Manager.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

GetSecretValue

En el siguiente ejemplo de código, se muestra cómo usar `GetSecretValue`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetSecretValue](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Amazon SES usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x con Amazon SES.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

ListIdentities

En el siguiente ejemplo de código, se muestra cómo usar `ListIdentities`.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListIdentities](#) la Referencia AWS SDK for Java 2.x de la API.

ListTemplates

En el siguiente ejemplo de código, se muestra cómo usar ListTemplates.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListTemplates](#) la Referencia AWS SDK for Java 2.x de la API.

SendEmail

En el siguiente ejemplo de código, se muestra cómo usar `SendEmail`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [SendEmail](#) la Referencia AWS SDK for Java 2.x de la API.

SendTemplatedEmail

En el siguiente ejemplo de código, se muestra cómo usar `SendTemplatedEmail`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [SendTemplatedEmail](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de SES API v2 usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK for Java 2.x API v2 de Amazon SES.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateContact

En el siguiente ejemplo de código, se muestra cómo usar CreateContact.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
try {
    // Create a new contact with the provided email address in the
    CreateContactRequest contactRequest = CreateContactRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .emailAddress(emailAddress)
        .build();

    sesClient.createContact(contactRequest);
    contacts.add(emailAddress);

    System.out.println("Contact created: " + emailAddress);

    // Send a welcome email to the new contact
    String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
    String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

    SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
        .fromEmailAddress(this.verifiedEmail)
        .destination(Destination.builder().toAddresses(emailAddress).build())
        .content(EmailContent.builder()
            .simple(
                Message.builder()
                    .subject(Content.builder().data("Welcome to the Weekly
Coupons Newsletter").build())
                    .body(Body.builder()
                        .text(Content.builder().data(welcomeText).build())
                        .html(Content.builder().data(welcomeHtml).build())
```

```
                .build())
            .build())
        .build())
    .build();
    SendEmailResponse welcomeEmailResponse =
sesClient.sendEmail(welcomeEmailRequest);
    System.out.println("Welcome email sent: " +
welcomeEmailResponse.messageId());
    } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
    } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
emailAddress + ": " + e.getMessage());
        throw e;
    }
}
```

- Para obtener más información sobre la API, consulta [CreateContact](#) la Referencia AWS SDK for Java 2.x de la API.

CreateContactList

En el siguiente ejemplo de código, se muestra cómo usar CreateContactList.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
try {
    // 2. Create a contact list
    String contactListName = CONTACT_LIST_NAME;
    CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
```



```
        .contactListName(contactListName)
        .build();
    sesClient.createContactList(createContactListRequest);
    System.out.println("Contact list created: " + contactListName);
} catch (AlreadyExistsException e) {
    System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
} catch (LimitExceededException e) {
    System.err.println("Limit for contact lists has been exceeded.");
    throw e;
} catch (SesV2Exception e) {
    System.err.println("Error creating contact list: " + e.getMessage());
    throw e;
}
```

- Para obtener más información sobre la API, consulta [CreateContactList](#) la Referencia AWS SDK for Java 2.x de la API.

CreateEmailIdentity

En el siguiente ejemplo de código, se muestra cómo usar CreateEmailIdentity.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
try {
    CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
    .emailIdentity(verifiedEmail)
    .build();
    sesClient.createEmailIdentity(createEmailIdentityRequest);
    System.out.println("Email identity created: " + verifiedEmail);
} catch (AlreadyExistsException e) {
    System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
}
```

```
    } catch (NotFoundException e) {
        System.err.println("The provided email address is not verified: " +
verifiedEmail);
        throw e;
    } catch (LimitExceededException e) {
        System.err
            .println("You have reached the limit for email identities. Please remove
some identities and try again.");
        throw e;
    } catch (SesV2Exception e) {
        System.err.println("Error creating email identity: " + e.getMessage());
        throw e;
    }
}
```

- Para obtener más información sobre la API, consulta [CreateEmailIdentity](#) la Referencia AWS SDK for Java 2.x de la API.

CreateEmailTemplate

En el siguiente ejemplo de código, se muestra cómo usar CreateEmailTemplate.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
try {
    // Create an email template named "weekly-coupons"
    String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
    String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

    CreateEmailTemplateRequest templateRequest =
CreateEmailTemplateRequest.builder()
        .templateName(TEMPLATE_NAME)
        .templateContent(EmailTemplateContent.builder()
```

```
        .subject("Weekly Coupons Newsletter")
        .html(newsletterHtml)
        .text(newsletterText)
        .build()
    .build();

    sesClient.createEmailTemplate(templateRequest);

    System.out.println("Email template created: " + TEMPLATE_NAME);
} catch (AlreadyExistsException e) {
    // If the template already exists, skip this step and proceed with the next
    // operation
    System.out.println("Email template already exists, skipping creation...");
} catch (LimitExceededException e) {
    // If the limit for email templates is exceeded, fail the workflow and inform
    // the user
    System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
    throw e;
} catch (Exception e) {
    System.err.println("Error occurred while creating email template: " +
e.getMessage());
    throw e;
}
```

- Para obtener más información sobre la API, consulta [CreateEmailTemplate](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteContactList

En el siguiente ejemplo de código, se muestra cómo usar DeleteContactList.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
try {
    // Delete the contact list
    DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .build();

    sesClient.deleteContactList(deleteContactListRequest);

    System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
} catch (NotFoundException e) {
    // If the contact list does not exist, log the error and proceed
    System.out.println("Contact list not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
    e.printStackTrace();
}
```

- Para obtener más información sobre la API, consulta [DeleteContactList](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteEmailIdentity

En el siguiente ejemplo de código, se muestra cómo usar DeleteEmailIdentity.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
try {
    // Delete the email identity
    DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
    .emailIdentity(this.verifiedEmail)
    .build();
```

```
sesClient.deleteEmailIdentity(deleteIdentityRequest);

System.out.println("Email identity deleted: " + this.verifiedEmail);
} catch (NotFoundException e) {
    // If the email identity does not exist, log the error and proceed
    System.out.println("Email identity not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
    e.printStackTrace();
}
} else {
    System.out.println("Skipping email identity deletion.");
}
}
```

- Para obtener más información sobre la API, consulta [DeleteEmailIdentity](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteEmailTemplate

En el siguiente ejemplo de código, se muestra cómo usar DeleteEmailTemplate.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
try {
    // Delete the template
    DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
    .templateName(TEMPLATE_NAME)
    .build();

    sesClient.deleteEmailTemplate(deleteTemplateRequest);
}
```

```
        System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
        // If the email template does not exist, log the error and proceed
        System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
        System.err.println("Error occurred while deleting the email template: " +
            e.getMessage());
        e.printStackTrace();
    }
}
```

- Para obtener más información sobre la API, consulta [DeleteEmailTemplate](#) la Referencia AWS SDK for Java 2.x de la API.

ListContacts

En el siguiente ejemplo de código, se muestra cómo usar ListContacts.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
ListContactsRequest contactListRequest = ListContactsRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .build();

List<String> contactEmails;
try {
    ListContactsResponse contactListResponse =
sesClient.listContacts(contactListRequest);

    contactEmails = contactListResponse.contacts().stream()
        .map(Contact::emailAddress)
        .toList();
} catch (Exception e) {
    // TODO: Remove when listContacts's GET body issue is resolved.
    contactEmails = this.contacts;
```

```
}
```

- Para obtener más información sobre la API, consulta [ListContacts](#) la Referencia AWS SDK for Java 2.x de la API.

SendEmail

En el siguiente ejemplo de código, se muestra cómo usar `SendEmail`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Envía un mensaje.

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

Envía un mensaje mediante una plantilla.

```
String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
for (String emailAddress : contactEmails) {
    SendEmailRequest newsletterRequest = SendEmailRequest.builder()
        .destination(Destination.builder().toAddresses(emailAddress).build())
        .content(EmailContent.builder()
            .template(Template.builder()
                .templateName(TEMPLATE_NAME)
                .templateData(coupons)
                .build())
            .build())
        .fromEmailAddress(this.verifiedEmail)
        .listManagementOptions(ListManagementOptions.builder()
            .contactListName(CONTACT_LIST_NAME)
            .build())
        .build();
    SendEmailResponse newsletterResponse =
sesClient.sendEmail(newsletterRequest);
    System.out.println("Newsletter sent to " + emailAddress + ": " +
newsletterResponse.messageId());
}
```

- Para obtener más información sobre la API, consulte [SendEmail](#) en la Referencia AWS SDK for Java 2.x de la API.

Escenarios

flujo de trabajo del boletín

El siguiente ejemplo de código muestra cómo usar el flujo de trabajo de los boletines de Amazon SES API v2.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
try {
    // 2. Create a contact list
    String contactListName = CONTACT_LIST_NAME;
    CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
        .contactListName(contactListName)
        .build();
    sesClient.createContactList(createContactListRequest);
    System.out.println("Contact list created: " + contactListName);
} catch (AlreadyExistsException e) {
    System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
} catch (LimitExceededException e) {
    System.err.println("Limit for contact lists has been exceeded.");
    throw e;
} catch (SesV2Exception e) {
    System.err.println("Error creating contact list: " + e.getMessage());
    throw e;
}

try {
    // Create a new contact with the provided email address in the
    CreateContactRequest contactRequest = CreateContactRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .emailAddress(emailAddress)
        .build();

    sesClient.createContact(contactRequest);
    contacts.add(emailAddress);

    System.out.println("Contact created: " + emailAddress);

    // Send a welcome email to the new contact
    String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
    String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

    SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
        .fromEmailAddress(this.verifiedEmail)
        .destination(Destination.builder().toAddresses(emailAddress).build())
        .content(EmailContent.builder()
            .simple(
```

```
        Message.builder()
            .subject(Content.builder().data("Welcome to the Weekly
Coupons Newsletter").build())
            .body(Body.builder()
                .text(Content.builder().data(welcomeText).build())
                .html(Content.builder().data(welcomeHtml).build())
                .build())
            .build()
        .build();
        SendEmailResponse welcomeEmailResponse =
sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
welcomeEmailResponse.messageId());
    } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
    } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
emailAddress + ": " + e.getMessage());
        throw e;
    }
}

ListContactsRequest contactListRequest = ListContactsRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .build();

List<String> contactEmails;
try {
    ListContactsResponse contactListResponse =
sesClient.listContacts(contactListRequest);

    contactEmails = contactListResponse.contacts().stream()
        .map(Contact::emailAddress)
        .toList();
} catch (Exception e) {
    // TODO: Remove when listContacts's GET body issue is resolved.
    contactEmails = this.contacts;
}
```

```
String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
for (String emailAddress : contactEmails) {
    SendEmailRequest newsletterRequest = SendEmailRequest.builder()
        .destination(Destination.builder().toAddresses(emailAddress).build())
        .content(EmailContent.builder()
            .template(Template.builder()
                .templateName(TEMPLATE_NAME)
                .templateData(coupons)
                .build())
            .build())
        .fromEmailAddress(this.verifiedEmail)
        .listManagementOptions(ListManagementOptions.builder()
            .contactListName(CONTACT_LIST_NAME)
            .build())
        .build();
    SendEmailResponse newsletterResponse =
sesClient.sendEmail(newsletterRequest);
    System.out.println("Newsletter sent to " + emailAddress + ": " +
newsletterResponse.messageId());
}

try {
    CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
        .emailIdentity(verifiedEmail)
        .build();
    sesClient.createEmailIdentity(createEmailIdentityRequest);
    System.out.println("Email identity created: " + verifiedEmail);
} catch (AlreadyExistsException e) {
    System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
} catch (NotFoundException e) {
    System.err.println("The provided email address is not verified: " +
verifiedEmail);
    throw e;
} catch (LimitExceededException e) {
    System.err
        .println("You have reached the limit for email identities. Please remove
some identities and try again.");
    throw e;
} catch (SesV2Exception e) {
    System.err.println("Error creating email identity: " + e.getMessage());
    throw e;
}
```

```
}

try {
    // Create an email template named "weekly-coupons"
    String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
    String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

    CreateEmailTemplateRequest templateRequest =
CreateEmailTemplateRequest.builder()
        .templateName(TEMPLATE_NAME)
        .templateContent(EmailTemplateContent.builder()
            .subject("Weekly Coupons Newsletter")
            .html(newsletterHtml)
            .text(newsletterText)
            .build())
        .build();

    sesClient.createEmailTemplate(templateRequest);

    System.out.println("Email template created: " + TEMPLATE_NAME);
} catch (AlreadyExistsException e) {
    // If the template already exists, skip this step and proceed with the next
    // operation
    System.out.println("Email template already exists, skipping creation...");
} catch (LimitExceededException e) {
    // If the limit for email templates is exceeded, fail the workflow and inform
    // the user
    System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
    throw e;
} catch (Exception e) {
    System.err.println("Error occurred while creating email template: " +
e.getMessage());
    throw e;
}

try {
    // Delete the contact list
    DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .build();
```

```
sesClient.deleteContactList(deleteContactListRequest);

System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
} catch (NotFoundException e) {
    // If the contact list does not exist, log the error and proceed
    System.out.println("Contact list not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
    e.printStackTrace();
}

try {
    // Delete the email identity
    DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
        .emailIdentity(this.verifiedEmail)
        .build();

    sesClient.deleteEmailIdentity(deleteIdentityRequest);

    System.out.println("Email identity deleted: " + this.verifiedEmail);
} catch (NotFoundException e) {
    // If the email identity does not exist, log the error and proceed
    System.out.println("Email identity not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
    e.printStackTrace();
}
} else {
    System.out.println("Skipping email identity deletion.");
}

try {
    // Delete the template
    DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
        .templateName(TEMPLATE_NAME)
        .build();

    sesClient.deleteEmailTemplate(deleteTemplateRequest);
```

```
        System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
        // If the email template does not exist, log the error and proceed
        System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
        System.err.println("Error occurred while deleting the email template: " +
            e.getMessage());
        e.printStackTrace();
    }
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateContact](#)
 - [CreateContactList](#)
 - [CreateEmailIdentity](#)
 - [CreateEmailTemplate](#)
 - [DeleteContactList](#)
 - [DeleteEmailIdentity](#)
 - [DeleteEmailTemplate](#)
 - [ListContacts](#)
 - [SendEmail.simple](#)
 - [SendEmail.plantilla](#)

Ejemplos de Amazon SNS usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Amazon SNS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Amazon SNS

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon SNS.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListTopics](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)
- [Ejemplos sin servidor](#)

Acciones

CheckIfPhoneNumberIsOptedOut

En el siguiente ejemplo de código, se muestra cómo usar `CheckIfPhoneNumberIsOptedOut`.

SDK para Java 2.x

Note

Hay más información al respecto en [GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CheckIfPhoneNumberIsOptedOut](#) la Referencia AWS SDK for Java 2.x de la API.

ConfirmSubscription

En el siguiente ejemplo de código, se muestra cómo usar `ConfirmSubscription`.

SDK para Java 2.x

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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 obtener más información sobre la API, consulta [ConfirmSubscription](#) la Referencia AWS SDK for Java 2.x de la API.

CreateTopic

En el siguiente ejemplo de código, se muestra cómo usar CreateTopic.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [CreateTopic](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteTopic

En el siguiente ejemplo de código, se muestra cómo usar DeleteTopic.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteTopic](#) la Referencia AWS SDK for Java 2.x de la API.

GetSMSAttributes

En el siguiente ejemplo de código, se muestra cómo usar GetSMSAttributes.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 ver la información de la API, consulte [GetSMSAttributes](#) en la Referencia de la API de AWS SDK for Java 2.x .

GetTopicAttributes

En el siguiente ejemplo de código, se muestra cómo usar `GetTopicAttributes`.

SDK para Java 2.x

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [GetTopicAttributes](#) la Referencia AWS SDK for Java 2.x de la API.

ListPhoneNumbersOptedOut

En el siguiente ejemplo de código, se muestra cómo usar `ListPhoneNumbersOptedOut`.

SDK para Java 2.x

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListPhoneNumbersOptedOut](#) la Referencia AWS SDK for Java 2.x de la API.

ListSubscriptions

En el siguiente ejemplo de código, se muestra cómo usar `ListSubscriptions`.

SDK para Java 2.x

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListSubscriptions](#) la Referencia AWS SDK for Java 2.x de la API.

ListTopics

En el siguiente ejemplo de código, se muestra cómo usar `ListTopics`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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"
            + "\nTopics\n\n" + result.topics());

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

- Para obtener más información sobre la API, consulta [ListTopics](#) la Referencia AWS SDK for Java 2.x de la API.

Publish

En el siguiente ejemplo de código, se muestra cómo usar Publish.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener información sobre la API, consulte [Publicar](#) en la Referencia de la API de AWS SDK for Java 2.x .

SetSMSAttributes

En el siguiente ejemplo de código, se muestra cómo usar SetSMSAttributes.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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();
        setSMSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSMSAttributes(SnsClient snsClient, HashMap<String, String>
attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
was "
                + result.sdkHttpResponse().statusCode());

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

```

- Para obtener información sobre la API, consulte [SetSMSAttributes](#) en la Referencia de la API de AWS SDK for Java 2.x .

SetSubscriptionAttributes

En el siguiente ejemplo de código, se muestra cómo usar `SetSubscriptionAttributes`.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [SetSubscriptionAttributes](#) la Referencia AWS SDK for Java 2.x de la API.

SetTopicAttributes

En el siguiente ejemplo de código, se muestra cómo usar SetTopicAttributes.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [SetTopicAttributes](#) la Referencia AWS SDK for Java 2.x de la API.

Subscribe

En el siguiente ejemplo de código, se muestra cómo usar `Subscribe`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Suscribe una dirección de correo electrónico a un tema.

```
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);
    }
}
}
}

```

Suscribe un punto final HTTP a un tema.

```

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);
    }
}
}
}

```

Suscriba una función Lambda a un tema.

```

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 obtener información sobre la API, consulte [Suscríbese](#) en la Referencia de la API de AWS SDK for Java 2.x .

TagResource

En el siguiente ejemplo de código, se muestra cómo usar TagResource.

SDK para Java 2.x

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [TagResource](#) la Referencia AWS SDK for Java 2.x de la API.

Unsubscribe

En el siguiente ejemplo de código, se muestra cómo usar Unsubscribe.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener información sobre la API, consulte [Cancelar suscripción](#) en la Referencia de la API de AWS SDK for Java 2.x .

Escenarios

Creación de un punto de enlace de la plataforma para notificaciones push

El siguiente ejemplo de código indica cómo crear un punto de enlace de la plataforma para las notificaciones push de Amazon SNS.

SDK para Java 2.x

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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);
        }
    }
}
```

Creación y publicación en un tema FIFO

El siguiente ejemplo de código indica cómo crear y publicar en un tema FIFO de Amazon SNS.

SDK para Java 2.x

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Este ejemplo

- crea un tema FIFO de Amazon SNS, dos colas FIFO de Amazon SQS y una cola estándar.
- suscribe las colas al tema y publica un mensaje en el tema.

La [prueba](#) verifica la recepción del mensaje en cada cola. El [ejemplo completo](#) también muestra la adición de políticas de acceso y, al final, elimina los recursos.

```
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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateTopic](#)
 - [Publicación](#)
 - [Subscribe](#)

Publicación de mensajes SMS en un tema

En el siguiente ejemplo de código, se muestra cómo:

- Cree un tema de Amazon SNS.
- Suscriba los números de teléfono al tema.

- Publique mensajes SMS en el tema para que todos los números de teléfono suscritos reciban el mensaje a la vez.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree un tema y devuelva su 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 "";
}
}
}

```

Suscriba un punto de enlace a un tema.

```

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);
    }
}
}
}

```

Establezca atributos en el mensaje, como el ID del remitente, el precio máximo y su tipo. Los atributos de mensaje son opcionales.

```

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 un mensaje en un tema. El mensaje se envía a cada suscriptor.

```

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);
    }
}
}
```

Publicación de un mensaje SMS

En el siguiente ejemplo de código se muestra cómo publicar mensajes SMS mediante Amazon SNS.

SDK para Java 2.x

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener información sobre la API, consulte [Publicar](#) en la Referencia de la API de AWS SDK for Java 2.x .

Publicación de mensajes en colas

En el siguiente ejemplo de código, se muestra cómo:

- Crear un tema (FIFO o no FIFO).
- Suscribirse a varias colas al tema con la opción de aplicar un filtro.
- Publicar mensajes en el tema.
- Sondear las colas en busca de los mensajes recibidos.

SDK para Java 2.x

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
package com.example.sns;

import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.ArrayList;
```



```
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        // if (args.length != 1) {
        // System.out.println(usage);
        // System.exit(1);
    }
}
```

```
// }

SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)

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

SqsClient sqsClient = SqsClient.builder()
    .region(Region.US_EAST_1)

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

Scanner in = new Scanner(System.in);
String accountId = "814548047983";
String useFIFO;
String duplication = "n";
String topicName;
String deduplicationID = null;
String groupId = null;

String topicArn;
String sqsQueueName;
String sqsQueueUrl;
String sqsQueueArn;
String subscriptionArn;
boolean selectFIFO = false;

String message;
List<Message> messageList;
List<String> filterList = new ArrayList<>();
String msgAttValue = "";

System.out.println(DASHES);
System.out.println("Welcome to messaging with topics and queues.");
System.out.println("In this workflow, you will create an SNS topic and
subscribe an SQS queue to the topic.\n" +
    "You can select from several options for configuring the topic and
the subscriptions for the queue.\n" +
    "You can then post to the topic and see the results in the queue.");
System.out.println(DASHES);

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

```
System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
    "FIFO topics deliver messages in order and support deduplication and
message filtering.\n" +
    "Would you like to work with FIFO topics? (y/n)");
useFIFO = in.nextLine();
if (useFIFO.compareTo("y") == 0) {
    selectFIFO = true;
    System.out.println("You have selected FIFO");
    System.out.println(" Because you have chosen a FIFO topic, deduplication
is supported.\n" +
        "        Deduplication IDs are either set in the message or
automatically generated from content using a hash function.\n"
        +
        "        If a message is successfully published to an SNS FIFO
topic, any message published and determined to have the same deduplication ID,\n"
        +
        "        within the five-minute deduplication interval, is
accepted but not delivered.\n" +
        "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

    System.out.println(
        "Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)");
    duplication = in.nextLine();
    if (duplication.compareTo("y") == 0) {
        System.out.println("Please enter a group id value");
        groupId = in.nextLine();
    } else {
        System.out.println("Please enter deduplication Id value");
        deduplicationID = in.nextLine();
        System.out.println("Please enter a group id value");
        groupId = in.nextLine();
    }
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a topic.");
System.out.println("Enter a name for your SNS topic.");
topicName = in.nextLine();
if (selectFIFO) {
```

```
        System.out.println("Because you have selected a FIFO topic, '.fifo' must
be appended to the topic name.");
        topicName = topicName + ".fifo";
        System.out.println("The name of the topic is " + topicName);
        topicArn = createFIFO(snsClient, topicName, duplication);
        System.out.println("The ARN of the FIFO topic is " + topicArn);

    } else {
        System.out.println("The name of the topic is " + topicName);
        topicArn = createSNSTopic(snsClient, topicName);
        System.out.println("The ARN of the non-FIFO topic is " + topicArn);

    }

    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Create an SQS queue.");
    System.out.println("Enter a name for your SQS queue.");
    sqsQueueName = in.nextLine();
    if (selectFIFO) {
        sqsQueueName = sqsQueueName + ".fifo";
    }
    sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
    System.out.println("The queue URL is " + sqsQueueUrl);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Get the SQS queue ARN attribute.");
    sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
    System.out.println("The ARN of the new queue is " + sqsQueueArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Attach an IAM policy to the queue.");

    // Define the policy to use. Make sure that you change the REGION if you are
    // running this code
    // in a different region.
    String policy = "{\n" +
        "    \"Statement\": [\n" +
        "        {\n" +
        "            \"Effect\": \"Allow\", \n" +
        "            \"Principal\": {\n" +
        "                \"Service\": \"sns.amazonaws.com\" \n" +
```

```

        "        },\n" +
        "        \"Action\": \"sqs:SendMessage\", \n" +
        "        \"Resource\": \"arn:aws:sqs:us-east-1:\" +
accountId + ":" + sqsQueueName + "\", \n" +
        "        \"Condition\": {\n" +
        "        \"ArnEquals\": {\n" +
        "        \"aws:SourceArn\": \"arn:aws:sns:us-east-1:\" +
accountId + ":" + topicName + "\"\n" +
        "        }\n" +
        "        }\n" +
        "    }\n" +
        " ]\n";

setQueueAttr(sqsClient, sqsQueueUrl, policy);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Subscribe to the SQS queue.");
if (selectFIFO) {
    System.out.println(
        "If you add a filter to this subscription, then only the
filtered messages will be received in the queue.\n"
        +
        "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
        +
        "For this example, you can filter messages by a \"tone\"
attribute.");
    System.out.println("Would you like to filter messages for " +
sqsQueueName + "'s subscription to the topic "
        + topicName + "? (y/n)");
    String filterAns = in.nextLine();
    if (filterAns.compareTo("y") == 0) {
        boolean moreAns = false;
        System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
        System.out.println("1. cheerful");
        System.out.println("2. funny");
        System.out.println("3. serious");
        System.out.println("4. sincere");
        while (!moreAns) {
            System.out.println("Select a number or choose 0 to end.");
            String ans = in.nextLine();

```

```
        switch (ans) {
            case "1":
                filterList.add("cheerful");
                break;
            case "2":
                filterList.add("funny");
                break;
            case "3":
                filterList.add("serious");
                break;
            case "4":
                filterList.add("sincere");
                break;
            default:
                moreAns = true;
                break;
        }
    }
}

subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Publish a message to the topic.");
if (selectFIFO) {
    System.out.println("Would you like to add an attribute to this message?
(y/n)");
    String msgAns = in.nextLine();
    if (msgAns.compareTo("y") == 0) {
        System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
        System.out.println("1. cheerful");
        System.out.println("2. funny");
        System.out.println("3. serious");
        System.out.println("4. sincere");
        System.out.println("Select a number or choose 0 to end.");
        String ans = in.nextLine();
        switch (ans) {
            case "1":
                msgAttValue = "cheerful";
                break;
            case "2":
                msgAttValue = "funny";
```

```
        break;
    case "3":
        msgAttValue = "serious";
        break;
    default:
        msgAttValue = "sincere";
        break;
    }

    System.out.println("Selected value is " + msgAttValue);
}
System.out.println("Enter a message.");
message = in.nextLine();
pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);

} else {
    System.out.println("Enter a message.");
    message = in.nextLine();
    pubMessage(snsClient, message, topicArn);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Display the message. Press any key to continue.");
in.nextLine();
messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
for (Message mes : messageList) {
    System.out.println("Message Id: " + mes.messageId());
    System.out.println("Full Message: " + mes.body());
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Delete the received message. Press any key to
continue.");
in.nextLine();
deleteMessages(sqsClient, sqsQueueUrl, messageList);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
in.nextLine();
```

```
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

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

    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);
            System.out.println(queueName + " was successfully deleted.");
        }
    }
}
```



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

public static void unSub(SnsClient snsClient, String subscriptionArn) {
    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("Status was " + result.sdkHttpResponse().statusCode()
            + "\nSubscription was removed for " +
request.subscriptionArn());

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

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
    try {
        List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
        for (Message msg : messages) {
            DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                .id(msg.messageId())
                .build();

            entries.add(entry);
        }

        DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
            .queueUrl(queueUrl)
            .entries(entries)
            .build();

        sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
        System.out.println("The batch delete of messages was successful");
    }
}
```

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

public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
    try {
        if (msgAttValue.isEmpty()) {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .maxNumberOfMessages(5)
                .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();
        } else {
            // We know there are filters on the message.
            ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageAttributeName(msgAttValue) // Include other message
attributes if needed.
                .maxNumberOfMessages(5)
                .build();

            return sqsClient.receiveMessage(receiveRequest).messages();
        }
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static void pubMessage(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);
    }
}

public static void pubMessageFIFO(SnsClient snsClient,
    String message,
    String topicArn,
    String msgAttValue,
    String duplication,
    String groupId,
    String deduplicationID) {

    try {
        PublishRequest request;
        // Means the user did not choose to use a message attribute.
        if (msgAttValue.isEmpty()) {
            if (duplication.compareTo("y") == 0) {
                request = PublishRequest.builder()
                    .message(message)
                    .messageGroupId(groupId)
                    .topicArn(topicArn)
                    .build();
            } else {
                request = PublishRequest.builder()
                    .message(message)
                    .messageDeduplicationId(deduplicationID)
                    .messageGroupId(groupId)
                    .topicArn(topicArn)
                    .build();
            }
        }

        } else {
            Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
            messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
                .dataType("String")
```

```
        .stringValue("true")
        .build());

    if (duplication.compareTo("y") == 0) {
        request = PublishRequest.builder()
            .message(message)
            .messageGroupId(groupId)
            .topicArn(topicArn)
            .build();
    } else {
        // Create a publish request with the message and attributes.
        request = PublishRequest.builder()
            .topicArn(topicArn)
            .message(message)
            .messageDeduplicationId(deduplicationID)
            .messageGroupId(groupId)
            .messageAttributes(messageAttributes)
            .build();
    }
}

// Publish the message to the topic.
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);
}
}

// Subscribe to the SQS queue.
public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
    try {
        SubscribeRequest request;
        if (filterList.isEmpty()) {
            // No filter subscription is added.
            request = SubscribeRequest.builder()
                .protocol("sqs")
                .endpoint(queueArn)
                .returnSubscriptionArn(true)
```

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

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
            "with the subscription ARN " + result.subscriptionArn());
        return result.subscriptionArn();
    } else {
        request = SubscribeRequest.builder()
            .protocol("sqs")
            .endpoint(queueArn)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
            "with the subscription ARN " + result.subscriptionArn());

        String attributeName = "FilterPolicy";
        Gson gson = new Gson();
        String jsonString = "{\"tone\": []}";
        JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
        JsonArray toneArray = jsonObject.getAsJsonArray("tone");
        for (String value : filterList) {
            toneArray.add(new JsonPrimitive(value));
        }

        String updatedJsonString = gson.toJson(jsonObject);
        System.out.println(updatedJsonString);
        SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
            .subscriptionArn(result.subscriptionArn())
            .attributeName(attributeName)
            .attributeValue(updatedJsonString)
            .build();

        snsClient.setSubscriptionAttributes(attRequest);
        return result.subscriptionArn();
    }
} catch (SnsException e) {
```

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

// Attach a policy to the queue.
public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
    try {
        Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
        attrMap.put(QueueAttributeName.POLICY, policy);

        SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributes(attrMap)
            .build();

        sqsClient.setQueueAttributes(attributesRequest);
        System.out.println("The policy has been successfully attached.");

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

public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
    // 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())
```

```
        return queueAtt.getValue();

    return "";
}

public static String createQueue(SqsClient sqsClient, String queueName, Boolean
selectFIFO) {
    try {
        System.out.println("\nCreate Queue");
        if (selectFIFO) {
            Map<QueueAttributeName, String> attrs = new HashMap<>();
            attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .attributes(attrs)
                .build();

            sqsClient.createQueue(createQueueRequest);
            System.out.println("\nGet queue url");
            GetQueueUrlResponse getQueueUrlResponse = sqsClient
.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
        } else {
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();

            sqsClient.createQueue(createQueueRequest);
            System.out.println("\nGet queue url");
            GetQueueUrlResponse getQueueUrlResponse = sqsClient
.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
        }

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

public static 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 "";
}

public static String createFIFO(SnsClient snsClient, String topicName, String
duplication) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = new HashMap<>();
        if (duplication.compareTo("n") == 0) {
            topicAttributes.put("FifoTopic", "true");
            topicAttributes.put("ContentBasedDeduplication", "false");
        } else {
            topicAttributes.put("FifoTopic", "true");
            topicAttributes.put("ContentBasedDeduplication", "true");
        }

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
            .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        return response.topicArn();

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


- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publicación](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

Ejemplos sin servidor

Invocar una función de Lambda desde un desencadenador de Amazon SNS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de un tema de SNS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de SNS con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
// SPDX-License-Identifier: Apache-2.0
```

```
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);
        }
    }
}
```

Ejemplos de Amazon SQS usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Amazon SQS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Amazon SQS

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon SQS.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListQueues](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)
- [Ejemplos sin servidor](#)

Acciones

CreateQueue

En el siguiente ejemplo de código, se muestra cómo usar CreateQueue.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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());
        return getQueueUrlResponse.queueUrl();

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

public static void listQueues(SqsClient sqsClient) {

    System.out.println("\nList Queues");
    String prefix = "que";

    try {
        ListQueuesRequest listQueuesRequest =
            ListQueuesRequest.builder().queueNamePrefix(prefix).build();
        ListQueuesResponse listQueuesResponse =
            sqsClient.listQueues(listQueuesRequest);
        for (String url : listQueuesResponse.queueUrls()) {
```

```
        System.out.println(url);
    }

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

public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
    // List queues with filters
    String namePrefix = "queue";
    ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
        .queueNamePrefix(namePrefix)
        .build();

    ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
    System.out.println("Queue URLs with prefix: " + namePrefix);
    for (String url : listQueuesFilteredResponse.queueUrls()) {
        System.out.println(url);
    }

    System.out.println("\nSend message");
    try {
        sqsClient.sendMessage(SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody("Hello world!")
            .delaySeconds(10)
            .build());

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

public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

    System.out.println("\nSend multiple messages");
    try {
        SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
            .queueUrl(queueUrl)
```

```
.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
                .build())
        .build();
    sqsClient.sendMessageBatch(sendMessageBatchRequest);

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

public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {

    System.out.println("\nReceive messages");
    try {
        ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .maxNumberOfMessages(5)
                .build();
        return sqsClient.receiveMessage(receiveMessageRequest).messages();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

    System.out.println("\nChange Message Visibility");
    try {

        for (Message message : messages) {
            ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
```



```
        .queueUrl(queueUrl)
        .receiptHandle(message.receiptHandle())
        .visibilityTimeout(100)
        .build();
    sqsClient.changeMessageVisibility(req);
}

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

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
    System.out.println("\nDelete Messages");

    try {
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
            sqsClient.deleteMessage(deleteMessageRequest);
        }
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Para obtener más información sobre la API, consulta [CreateQueue](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteMessage

En el siguiente ejemplo de código, se muestra cómo usar DeleteMessage.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteMessage](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteQueue

En el siguiente ejemplo de código, se muestra cómo usar DeleteQueue.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DeleteQueue](#) la Referencia AWS SDK for Java 2.x de la API.

GetQueueUrl

En el siguiente ejemplo de código, se muestra cómo usar `GetQueueUrl`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
        GetQueueUrlResponse getQueueUrlResponse = sqsClient
            .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        return getQueueUrlResponse.queueUrl();
```

- Para obtener más información sobre la API, consulta [GetQueueUrl](#) la Referencia AWS SDK for Java 2.x de la API.

ListQueues

En el siguiente ejemplo de código, se muestra cómo usar `ListQueues`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }

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

- Para obtener más información sobre la API, consulta [ListQueues](#) la Referencia AWS SDK for Java 2.x de la API.

ReceiveMessage

En el siguiente ejemplo de código, se muestra cómo usar `ReceiveMessage`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ReceiveMessage](#) la Referencia AWS SDK for Java 2.x de la API.

SendMessage

En el siguiente ejemplo de código, se muestra cómo usar SendMessage.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [SendMessage](#) la Referencia AWS SDK for Java 2.x de la API.

SendMessageBatch

En el siguiente ejemplo de código, se muestra cómo usar SendMessageBatch.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
```



```
        .queueUrl(queueUrl)

        .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)

                .build())
        .build();
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- Para obtener más información sobre la API, consulta [SendMessageBatch](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Creación y publicación en un tema FIFO

El siguiente ejemplo de código indica cómo crear y publicar en un tema FIFO de Amazon SNS.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Este ejemplo

- crea un tema FIFO de Amazon SNS, dos colas FIFO de Amazon SQS y una cola estándar.
- suscribe las colas al tema y publica un mensaje en el tema.

La [prueba](#) verifica la recepción del mensaje en cada cola. El [ejemplo completo](#) también muestra la adición de políticas de acceso y, al final, elimina los recursos.

```
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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateTopic](#)
 - [Publicación](#)
 - [Subscribe](#)

Publicación de mensajes en colas

En el siguiente ejemplo de código, se muestra cómo:

- Crear un tema (FIFO o no FIFO).
- Suscribirse varias colas al tema con la opción de aplicar un filtro.
- Publicar mensajes en el tema.
- Sondear las colas en busca de los mensajes recibidos.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
package com.example.sns;

import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
```

```
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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. Gives the user three options to choose from.
* 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
* 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
* 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
* 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
* 6. Subscribes to the SQS queue.
* 7. Publishes a message to the topic.
* 8. Displays the messages.
* 9. Deletes the received message.
* 10. Unsubscribes from the topic.
* 11. Deletes the SNS topic.
*/
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

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

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)

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

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)

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

        Scanner in = new Scanner(System.in);
        String accountId = "814548047983";
        String useFIFO;
        String duplication = "n";
        String topicName;
```

```
String deduplicationID = null;
String groupId = null;

String topicArn;
String sqsQueueName;
String sqsQueueUrl;
String sqsQueueArn;
String subscriptionArn;
boolean selectFIFO = false;

String message;
List<Message> messageList;
List<String> filterList = new ArrayList<>();
String msgAttValue = "";

System.out.println(DASHES);
System.out.println("Welcome to messaging with topics and queues.");
System.out.println("In this workflow, you will create an SNS topic and
subscribe an SQS queue to the topic.\n" +
    "You can select from several options for configuring the topic and
the subscriptions for the queue.\n" +
    "You can then post to the topic and see the results in the queue.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
    "FIFO topics deliver messages in order and support deduplication and
message filtering.\n" +
    "Would you like to work with FIFO topics? (y/n)");
useFIFO = in.nextLine();
if (useFIFO.compareTo("y") == 0) {
    selectFIFO = true;
    System.out.println("You have selected FIFO");
    System.out.println(" Because you have chosen a FIFO topic, deduplication
is supported.\n" +
        " Deduplication IDs are either set in the message or
automatically generated from content using a hash function.\n"
        +
        " If a message is successfully published to an SNS FIFO
topic, any message published and determined to have the same deduplication ID,\n"
        +
        " within the five-minute deduplication interval, is
accepted but not delivered.\n" +
```



```
        "        For more information about deduplication, see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

        System.out.println(
            "Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)");
        duplication = in.nextLine();
        if (duplication.compareTo("y") == 0) {
            System.out.println("Please enter a group id value");
            groupId = in.nextLine();
        } else {
            System.out.println("Please enter deduplication Id value");
            deduplicationID = in.nextLine();
            System.out.println("Please enter a group id value");
            groupId = in.nextLine();
        }
    }
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a topic.");
System.out.println("Enter a name for your SNS topic.");
topicName = in.nextLine();
if (selectFIFO) {
    System.out.println("Because you have selected a FIFO topic, '.fifo' must
be appended to the topic name.");
    topicName = topicName + ".fifo";
    System.out.println("The name of the topic is " + topicName);
    topicArn = createFIFO(snsClient, topicName, duplication);
    System.out.println("The ARN of the FIFO topic is " + topicArn);
} else {
    System.out.println("The name of the topic is " + topicName);
    topicArn = createSNSTopic(snsClient, topicName);
    System.out.println("The ARN of the non-FIFO topic is " + topicArn);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create an SQS queue.");
System.out.println("Enter a name for your SQS queue.");
sqsQueueName = in.nextLine();
if (selectFIFO) {
```

```

        sqsQueueName = sqsQueueName + ".fifo";
    }
    sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
    System.out.println("The queue URL is " + sqsQueueUrl);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Get the SQS queue ARN attribute.");
    sqsQueueArn = getSqsQueueAttrs(sqsClient, sqsQueueUrl);
    System.out.println("The ARN of the new queue is " + sqsQueueArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Attach an IAM policy to the queue.");

    // Define the policy to use. Make sure that you change the REGION if you are
    // running this code
    // in a different region.
    String policy = "{\n" +
        "    \"Statement\": [\n" +
        "        {\n" +
        "            \"Effect\": \"Allow\",\n" +
        "            \"Principal\": {\n" +
        "                \"Service\": \"sns.amazonaws.com\"\n" +
        "            },\n" +
        "            \"Action\": \"sqs:SendMessage\",\n" +
        "            \"Resource\": \"arn:aws:sqs:us-east-1:\" +
        accountId + ":" + sqsQueueName + "\",\n" +
        "            \"Condition\": {\n" +
        "                \"ArnEquals\": {\n" +
        "                    \"aws:SourceArn\": \"arn:aws:sns:us-east-1:\" +
        accountId + ":" + topicName + "\"\n" +
        "                }\n" +
        "            }\n" +
        "        }\n" +
        "    ]\n" +
        " }";

    setQueueAttr(sqsClient, sqsQueueUrl, policy);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Subscribe to the SQS queue.");
    if (selectFIFO) {

```

```
        System.out.println(
            "If you add a filter to this subscription, then only the
filtered messages will be received in the queue.\n"
            +
            "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
            +
            "For this example, you can filter messages by a \"tone\"
attribute.");
        System.out.println("Would you like to filter messages for " +
sqsQueueName + "'s subscription to the topic "
            + topicName + "? (y/n)");
        String filterAns = in.nextLine();
        if (filterAns.compareTo("y") == 0) {
            boolean moreAns = false;
            System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
            System.out.println("1. cheerful");
            System.out.println("2. funny");
            System.out.println("3. serious");
            System.out.println("4. sincere");
            while (!moreAns) {
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        filterList.add("cheerful");
                        break;
                    case "2":
                        filterList.add("funny");
                        break;
                    case "3":
                        filterList.add("serious");
                        break;
                    case "4":
                        filterList.add("sincere");
                        break;
                    default:
                        moreAns = true;
                        break;
                }
            }
        }
    }
}
```

```
subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Publish a message to the topic.");
if (selectFIFO) {
    System.out.println("Would you like to add an attribute to this message?
(y/n)");
    String msgAns = in.nextLine();
    if (msgAns.compareTo("y") == 0) {
        System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
        System.out.println("1. cheerful");
        System.out.println("2. funny");
        System.out.println("3. serious");
        System.out.println("4. sincere");
        System.out.println("Select a number or choose 0 to end.");
        String ans = in.nextLine();
        switch (ans) {
            case "1":
                msgAttValue = "cheerful";
                break;
            case "2":
                msgAttValue = "funny";
                break;
            case "3":
                msgAttValue = "serious";
                break;
            default:
                msgAttValue = "sincere";
                break;
        }

        System.out.println("Selected value is " + msgAttValue);
    }
    System.out.println("Enter a message.");
    message = in.nextLine();
    pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);
} else {
    System.out.println("Enter a message.");
    message = in.nextLine();
    pubMessage(snsClient, message, topicArn);
}
```

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

    System.out.println(DASHES);
    System.out.println("8. Display the message. Press any key to continue.");
    in.nextLine();
    messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
    for (Message mes : messageList) {
        System.out.println("Message Id: " + mes.messageId());
        System.out.println("Full Message: " + mes.body());
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("9. Delete the received message. Press any key to
continue.");
    in.nextLine();
    deleteMessages(sqsClient, sqsQueueUrl, messageList);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
    in.nextLine();
    unSub(snsClient, subscriptionArn);
    deleteSQSQueue(sqsClient, sqsQueueName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("11. Delete the topic. Press any key to continue.");
    in.nextLine();
    deleteSNSTopic(snsClient, topicArn);

    System.out.println(DASHES);
    System.out.println("The SNS/SQS workflow has completed successfully.");
    System.out.println(DASHES);
}

public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();
```

```
        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

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

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);
        System.out.println(queueName + " was successfully deleted.");

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

public static void unSub(SnsClient snsClient, String subscriptionArn) {
    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("Status was " + result.sdkHttpResponse().statusCode()
            + "\nSubscription was removed for " +
request.subscriptionArn());

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

```
    }  
  }  
  
  public static void deleteMessages(SqsClient sqsClient, String queueUrl,  
  List<Message> messages) {  
    try {  
      List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();  
      for (Message msg : messages) {  
        DeleteMessageBatchRequestEntry entry =  
DeleteMessageBatchRequestEntry.builder()  
          .id(msg.messageId())  
          .build();  
  
        entries.add(entry);  
      }  
  
      DeleteMessageBatchRequest deleteMessageBatchRequest =  
DeleteMessageBatchRequest.builder()  
        .queueUrl(queueUrl)  
        .entries(entries)  
        .build();  
  
      sqsClient.deleteMessageBatch(deleteMessageBatchRequest);  
      System.out.println("The batch delete of messages was successful");  
  
    } catch (SqsException e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
      System.exit(1);  
    }  
  }  
  
  public static List<Message> receiveMessages(SqsClient sqsClient, String  
  queueUrl, String msgAttValue) {  
    try {  
      if (msgAttValue.isEmpty()) {  
        ReceiveMessageRequest receiveMessageRequest =  
ReceiveMessageRequest.builder()  
          .queueUrl(queueUrl)  
          .maxNumberOfMessages(5)  
          .build();  
        return sqsClient.receiveMessage(receiveMessageRequest).messages();  
      } else {  
        // We know there are filters on the message.  

```

```
        ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
        .maxNumberOfMessages(5)
        .build();

        return sqsClient.receiveMessage(receiveRequest).messages();
    }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static void pubMessage(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);
    }
}

public static void pubMessageFIFO(SnsClient snsClient,
    String message,
    String topicArn,
    String msgAttValue,
    String duplication,
    String groupId,
    String deduplicationID) {
```



```
try {
    PublishRequest request;
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            request = PublishRequest.builder()
                .message(message)
                .messageGroupId(groupId)
                .topicArn(topicArn)
                .build();
        } else {
            request = PublishRequest.builder()
                .message(message)
                .messageDeduplicationId(deduplicationID)
                .messageGroupId(groupId)
                .topicArn(topicArn)
                .build();
        }
    }

    } else {
        Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
        messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
            .dataType("String")
            .stringValue("true")
            .build());

        if (duplication.compareTo("y") == 0) {
            request = PublishRequest.builder()
                .message(message)
                .messageGroupId(groupId)
                .topicArn(topicArn)
                .build();
        } else {
            // Create a publish request with the message and attributes.
            request = PublishRequest.builder()
                .topicArn(topicArn)
                .message(message)
                .messageDeduplicationId(deduplicationID)
                .messageGroupId(groupId)
                .messageAttributes(messageAttributes)
                .build();
        }
    }
}
```

```
    }

    // Publish the message to the topic.
    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);
    }
}

// Subscribe to the SQS queue.
public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
    try {
        SubscribeRequest request;
        if (filterList.isEmpty()) {
            // No filter subscription is added.
            request = SubscribeRequest.builder()
                .protocol("sqs")
                .endpoint(queueArn)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                "with the subscription ARN " + result.subscriptionArn());
            return result.subscriptionArn();
        } else {
            request = SubscribeRequest.builder()
                .protocol("sqs")
                .endpoint(queueArn)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
```

```
        "with the subscription ARN " + result.subscriptionArn());

    String attributeName = "FilterPolicy";
    Gson gson = new Gson();
    String jsonString = "{\"tone\": []}";
    JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
    JsonArray toneArray = jsonObject.getAsJsonArray("tone");
    for (String value : filterList) {
        toneArray.add(new JsonPrimitive(value));
    }

    String updatedJsonString = gson.toJson(jsonObject);
    System.out.println(updatedJsonString);
    SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
        .subscriptionArn(result.subscriptionArn())
        .attributeName(attributeName)
        .attributeValue(updatedJsonString)
        .build();

    snsClient.setSubscriptionAttributes(attRequest);
    return result.subscriptionArn();
}

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

// Attach a policy to the queue.
public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
    try {
        Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
        attrMap.put(QueueAttributeName.POLICY, policy);

        SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributes(attrMap)
            .build();
```

```
        sqsClient.setQueueAttributes(attributesRequest);
        System.out.println("The policy has been successfully attached.");

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

public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
    // 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())
        return queueAtt.getValue();

    return "";
}

public static String createQueue(SqsClient sqsClient, String queueName, Boolean
selectFIFO) {
    try {
        System.out.println("\nCreate Queue");
        if (selectFIFO) {
            Map<QueueAttributeName, String> attrs = new HashMap<>();
            attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .attributes(attrs)
                .build();

            sqsClient.createQueue(createQueueRequest);
            System.out.println("\nGet queue url");
```

```
        GetQueueUrlResponse getQueueUrlResponse = sqsClient
.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        return getQueueUrlResponse.queueUrl();
    } else {
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .build();

        sqsClient.createQueue(createQueueRequest);
        System.out.println("\nGet queue url");
        GetQueueUrlResponse getQueueUrlResponse = sqsClient
.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        return getQueueUrlResponse.queueUrl();
    }

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

public static 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 "";
}

public static String createFIFO(SnsClient snsClient, String topicName, String
duplication) {
    try {
```

```
// Create a FIFO topic by using the SNS service client.
Map<String, String> topicAttributes = new HashMap<>();
if (duplication.compareTo("n") == 0) {
    topicAttributes.put("FifoTopic", "true");
    topicAttributes.put("ContentBasedDeduplication", "false");
} else {
    topicAttributes.put("FifoTopic", "true");
    topicAttributes.put("ContentBasedDeduplication", "true");
}

CreateTopicRequest topicRequest = CreateTopicRequest.builder()
    .name(topicName)
    .attributes(topicAttributes)
    .build();

CreateTopicResponse response = snsClient.createTopic(topicRequest);
return response.topicArn();

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

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publicación](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)

- [Unsubscribe](#)

Ejemplos sin servidor

Invocar una función de Lambda desde un desencadenador de Amazon SQS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de una cola de SQS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de SQS con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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;
    }


}

}
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Amazon SQS.

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de una cola de SQS. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Java 2.x

 Note

Hay más información [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Notificación de los errores de los elementos del lote de SQS con Lambda mediante Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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);  
}  
}
```

Ejemplos de funciones de pasos usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Java 2.x with Step Functions.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Step Functions

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Step Functions.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Versión Java de Introducción.

```
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 obtener más información sobre la API, consulta [ListStateMachines](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateActivity

En el siguiente ejemplo de código, se muestra cómo usar CreateActivity.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();
```

```
        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();

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

- Para obtener más información sobre la API, consulta [CreateActivity](#) la Referencia AWS SDK for Java 2.x de la API.

CreateStateMachine

En el siguiente ejemplo de código, se muestra cómo usar CreateStateMachine.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
            .roleArn(roleARN)
            .type(StateMachineType.STANDARD)
            .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();
    }
```

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

- Para obtener más información sobre la API, consulta [CreateStateMachine](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteActivity

En el siguiente ejemplo de código, se muestra cómo usar DeleteActivity.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteActivity(SfnClient sfnClient, String actArn) {  
    try {  
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()  
            .activityArn(actArn)  
            .build();  
  
        sfnClient.deleteActivity(activityRequest);  
        System.out.println("You have deleted " + actArn);  
  
    } catch (SfnException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obtener más información sobre la API, consulta [DeleteActivity](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteStateMachine

En el siguiente ejemplo de código, se muestra cómo usar DeleteStateMachine.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        while (true) {
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
            System.out.println("The state machine is not deleted yet. The status
is " + response.status());
            Thread.sleep(3000);
        }

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(stateMachineArn + " was successfully deleted.");
}
```

- Para obtener más información sobre la API, consulta [DeleteStateMachine](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeExecution

En el siguiente ejemplo de código, se muestra cómo usar DescribeExecution.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeExe(SfnClient sfnClient, String executionArn) {
    try {
        DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
            .executionArn(executionArn)
            .build();

        String status = "";
        boolean hasSucceeded = false;
        while (!hasSucceeded) {
            DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
            status = response.statusAsString();
            if (status.compareTo("RUNNING") == 0) {
                System.out.println("The state machine is still running, let's
wait for it to finish.");
                Thread.sleep(2000);
            } else if (status.compareTo("SUCCEEDED") == 0) {
                System.out.println("The Step Function workflow has succeeded");
                hasSucceeded = true;
            } else {
                System.out.println("The Status is neither running or
succeeded");
            }
        }
    }
}
```

```
    }
    System.out.println("The Status is " + status);

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

- Para obtener más información sobre la API, consulta [DescribeExecution](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeStateMachine

En el siguiente ejemplo de código, se muestra cómo usar DescribeStateMachine.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
    try {
        DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
        System.out.println("The name of the State machine is " +
response.name());
        System.out.println("The status of the State machine is " +
response.status());
        System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
    }
}
```



```
        System.out.println("The role ARN value is " + response.roleArn());
    } catch (SfnException e) {
        System.err.println(e.getMessage());
    }
}
```

- Para obtener más información sobre la API, consulta [DescribeStateMachine](#) la Referencia AWS SDK for Java 2.x de la API.

GetActivityTask

En el siguiente ejemplo de código, se muestra cómo usar `GetActivityTask`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
    GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
    sfnClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

/// <summary>
/// Stop execution of a Step Functions workflow.
/// </summary>
/// <param name="executionArn">The Amazon Resource Name (ARN) of
```

```
/// the Step Functions execution to stop.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> StopExecution(string executionArn)
{
    var response =
        await _amazonStepFunctions.StopExecutionAsync(new StopExecutionRequest
        { ExecutionArn = executionArn });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- Para obtener más información sobre la API, consulta [GetActivityTask](#) la Referencia AWS SDK for Java 2.x de la API.

ListActivities

En el siguiente ejemplo de código, se muestra cómo usar `ListActivities`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListActivities](#) la Referencia AWS SDK for Java 2.x de la API.

ListExecutions

En el siguiente ejemplo de código, se muestra cómo usar `ListExecutions`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void getExeHistory(SfnClient sfnClient, String exeARN) {
    try {
        GetExecutionHistoryRequest historyRequest =
        GetExecutionHistoryRequest.builder()
            .executionArn(exeARN)
            .maxResults(10)
            .build();

        GetExecutionHistoryResponse historyResponse =
        sfnClient.getExecutionHistory(historyRequest);
        List<HistoryEvent> events = historyResponse.events();
        for (HistoryEvent event : events) {
            System.out.println("The event type is " + event.type().toString());
        }

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

- Para obtener más información sobre la API, consulta [ListExecutions](#) la Referencia AWS SDK for Java 2.x de la API.

ListStateMachines

En el siguiente ejemplo de código, se muestra cómo usar `ListStateMachines`.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListStateMachines](#) la Referencia AWS SDK for Java 2.x de la API.

SendTaskSuccess

En el siguiente ejemplo de código, se muestra cómo usar `SendTaskSuccess`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);

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

```
}
```

- Para obtener más información sobre la API, consulta [SendTaskSuccess](#) la Referencia AWS SDK for Java 2.x de la API.

StartExecution

En el siguiente ejemplo de código, se muestra cómo usar `StartExecution`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(jsonEx)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
        return response.executionArn();

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

- Para obtener más información sobre la API, consulta [StartExecution](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Introducción a las máquinas de estado

En el siguiente ejemplo de código, se muestra cómo:

- Crear una actividad
- Crear una máquina de estado a partir de una definición de Amazon States Language que contenga la actividad creada anteriormente como un paso
- Ejecutar la máquina de estados y responder a la actividad con entradas de usuario
- Obtener el resultado y el estado final una vez completada la ejecución y, luego, limpiar los recursos.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CreateActivity](#)
 - [CreateStateMachine](#)
 - [DeleteActivity](#)
 - [DeleteStateMachine](#)
 - [DescribeExecution](#)
 - [DescribeStateMachine](#)
 - [GetActivityTask](#)
 - [ListActivities](#)
 - [ListStateMachines](#)
 - [SendTaskSuccess](#)
 - [StartExecution](#)
 - [StopExecution](#)

AWS STS ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with AWS STS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

AssumeRole

En el siguiente ejemplo de código, se muestra cómo usar AssumeRole.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [AssumeRole](#) la Referencia AWS SDK for Java 2.x de la API.

AWS Support ejemplos de uso de SDK for Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Java 2.x with AWS Support.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

¿Hola AWS Support

En los siguientes ejemplos de código se muestra cómo empezar a utilizar AWS Glue.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeServices](#) la Referencia AWS SDK for Java 2.x de la API.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

AddAttachmentsToSet

En el siguiente ejemplo de código, se muestra cómo usar AddAttachmentsToSet.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
```

```
try {
    File myFile = new File(fileAttachment);
    InputStream sourceStream = new FileInputStream(myFile);
    SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

    Attachment attachment = Attachment.builder()
        .fileName(myFile.getName())
        .data(sourceBytes)
        .build();

    AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
        .attachments(attachment)
        .build();

    AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
    return response.attachmentSetId();

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

- Para obtener más información sobre la API, consulta [AddAttachmentsToSet](#) la Referencia AWS SDK for Java 2.x de la API.

AddCommunicationToCase

En el siguiente ejemplo de código, se muestra cómo usar AddCommunicationToCase.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


```
public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

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

- Para obtener más información sobre la API, consulta [AddCommunicationToCase](#) la Referencia AWS SDK for Java 2.x de la API.

CreateCase

En el siguiente ejemplo de código, se muestra cómo usar CreateCase.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with " +
serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
            .issueType("technical")
            .build();

        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();

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

- Para obtener más información sobre la API, consulta [CreateCase](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeAttachment

En el siguiente ejemplo de código, se muestra cómo usar DescribeAttachment.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is " +
response.attachment().fileName());

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

- Para obtener más información sobre la API, consulta [DescribeAttachment](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeCases

En el siguiente ejemplo de código, se muestra cómo usar DescribeCases.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);
```

```
DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
    .maxResults(20)
    .afterTime(yesterday.toString())
    .beforeTime(now.toString())
    .build();

DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
List<CaseDetails> cases = response.cases();
for (CaseDetails sinCase : cases) {
    System.out.println("The case status is " + sinCase.status());
    System.out.println("The case Id is " + sinCase.caseId());
    System.out.println("The case subject is " + sinCase.subject());
}

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

- Para obtener más información sobre la API, consulta [DescribeCases](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeCommunications

En el siguiente ejemplo de código, se muestra cómo usar DescribeCommunications.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String listCommunications(SupportClient supportClient, String
caseId) {
```

```
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
            .build();

        DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm : communications) {
            System.out.println("the body is: " + comm.body());

            // Get the attachment id value.
            List<AttachmentDetails> attachments = comm.attachmentSet();
            for (AttachmentDetails detail : attachments) {
                attachId = detail.attachmentId();
            }
        }
        return attachId;
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Para obtener más información sobre la API, consulta [DescribeCommunications](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeServices

En el siguiente ejemplo de código, se muestra cómo usar DescribeServices.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service : services) {
            if (index == 11)
                break;

            System.out.println("The Service name is: " + service.name());
            if (service.name().compareTo("Account") == 0)
                serviceCode = service.code();

            // Get the Categories for this service.
            List<Category> categories = service.categories();
            for (Category cat : categories) {
                System.out.println("The category name is: " + cat.name());
                if (cat.name().compareTo("Security") == 0)
                    catName = cat.name();
            }
            index++;
        }
    }
}
```

```
        // Push the two values to the list.
        sevCatList.add(serviceCode);
        sevCatList.add(catName);
        return sevCatList;

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

- Para obtener más información sobre la API, consulta [DescribeServices](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeSeverityLevels

En el siguiente ejemplo de código, se muestra cómo usar `DescribeSeverityLevels`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
            .language("en")
            .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel : severityLevels) {
```

```
        System.out.println("The severity level name is: " +
sevLevel.name());
        if (sevLevel.name().compareTo("High") == 0)
            levelName = sevLevel.name();
    }
    return levelName;

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

- Para obtener más información sobre la API, consulta [DescribeSeverityLevels](#) la Referencia AWS SDK for Java 2.x de la API.

ResolveCase

En el siguiente ejemplo de código, se muestra cómo usar ResolveCase.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());
    }
```



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

- Para obtener más información sobre la API, consulta [ResolveCase](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Introducción a los casos

En el siguiente ejemplo de código, se muestra cómo:

- Obtenga y muestre los servicios disponibles y los niveles de gravedad de los casos.
- Cree un caso de asistencia mediante un servicio, una categoría y un nivel de gravedad seleccionados.
- Obtenga y muestre una lista de casos abiertos para el día actual.
- Añada una serie de archivos adjuntos y una comunicación al nuevo caso.
- Describa el nuevo archivo adjunto y la comunicación del caso.
- Resuelva el caso.
- Obtenga y muestre una lista de casos resueltos para el día actual.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute varias AWS Support operaciones.

```
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.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                .maxResults(30)
                .afterTime(yesterday.toString())
                .beforeTime(now.toString())
                .includeResolvedCases(true)
                .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
```

```
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            if (sinCase.status().compareTo("resolved") == 0)
                System.out.println("The case status is " + sinCase.status());
        }

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

public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

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

public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is " +
response.attachment().fileName());

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

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

public static String listCommunications(SupportClient supportClient, String
caseId) {
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
            .build();

        DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm : communications) {
            System.out.println("the body is: " + comm.body());

            // Get the attachment id value.
            List<AttachmentDetails> attachments = comm.attachmentSet();
            for (AttachmentDetails detail : attachments) {
                attachId = detail.attachmentId();
            }
        }
        return attachId;
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();
    }
}
```

```
        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

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

public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
            .attachments(attachment)
            .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

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



```
public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            System.out.println("The case status is " + sinCase.status());
            System.out.println("The case Id is " + sinCase.caseId());
            System.out.println("The case subject is " + sinCase.subject());
        }

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

public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with " +
serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
            .issueType("technical")
            .build();
    }
}
```

```
        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();

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

public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
            .language("en")
            .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel : severityLevels) {
            System.out.println("The severity level name is: " +
sevLevel.name());
            if (sevLevel.name().compareTo("High") == 0)
                levelName = sevLevel.name();
        }
        return levelName;

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

// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();
```

```
        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service : services) {
            if (index == 11)
                break;

            System.out.println("The Service name is: " + service.name());
            if (service.name().compareTo("Account") == 0)
                serviceCode = service.code();

            // Get the Categories for this service.
            List<Category> categories = service.categories();
            for (Category cat : categories) {
                System.out.println("The category name is: " + cat.name());
                if (cat.name().compareTo("Security") == 0)
                    catName = cat.name();
            }
            index++;
        }

        // Push the two values to the list.
        sevCatList.add(serviceCode);
        sevCatList.add(catName);
        return sevCatList;

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

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
 - [AddAttachmentsToSet](#)
 - [AddCommunicationToCase](#)
 - [CreateCase](#)
 - [DescribeAttachment](#)
 - [DescribeCases](#)
 - [DescribeCommunications](#)
 - [DescribeServices](#)
 - [DescribeSeverityLevels](#)
 - [ResolveCase](#)

Ejemplos de Systems Manager usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x mediante Systems Manager.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Hola, Systems Manager

Los siguientes ejemplos de código muestran cómo empezar a utilizar Systems Manager.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DocumentFilter;
import software.amazon.awssdk.services.ssm.model.ListDocumentsRequest;
import software.amazon.awssdk.services.ssm.model.ListDocumentsResponse;

public class HelloSSM {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <awsAccount>

            Where:
                awsAccount - Your AWS Account number.
            """;

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

        String awsAccount = args[0] ;
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        listDocuments(ssmClient, awsAccount);
    }

    /*
```

```
This code automatically fetches the next set of results using the `nextToken`
and
stops once the desired maxResults (20 in this case) have been reached.
*/
public static void listDocuments(SsmClient ssmClient, String awsAccount) {
    String nextToken = null;
    int totalDocumentsReturned = 0;
    int maxResults = 20;
    do {
        ListDocumentsRequest request = ListDocumentsRequest.builder()
            .documentFilterList(
                DocumentFilter.builder()
                    .key("Owner")
                    .value(awsAccount)
                    .build()
            )
            .maxResults(maxResults)
            .nextToken(nextToken)
            .build();

        ListDocumentsResponse response = ssmClient.listDocuments(request);
        response.documentIdentifiers().forEach(identifier ->
System.out.println("Document Name: " + identifier.name()));
        nextToken = response.nextToken();
        totalDocumentsReturned += response.documentIdentifiers().size();
    } while (nextToken != null && totalDocumentsReturned < maxResults);
    }
}
```

- Para obtener información sobre la API, consulte [listThings](#) en la Referencia de la API de AWS SDK for Java 2.x .

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateDocument

En el siguiente ejemplo de código, se muestra cómo usar CreateDocument.

SDK para Java 2.x

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Create an AWS SSM document to use in this scenario.
public static void createSSMDoc(SsmClient ssmClient, String docName) {
    // Create JSON for the content
    String jsonData = ""
        {
            "schemaVersion": "2.2",
            "description": "Run a simple shell command",
            "mainSteps": [
                {
                    "action": "aws:runShellScript",
                    "name": "runEchoCommand",
                    "inputs": {
                        "runCommand": [
                            "echo 'Hello, world!'"
                        ]
                    }
                }
            ]
        }
        """;

    try {
        CreateDocumentRequest request = CreateDocumentRequest.builder()
            .content(jsonData)
            .name(docName)
            .documentType(DocumentType.COMMAND)
            .build();

        // Create the document.
```

```
        CreateDocumentResponse response = ssmClient.createDocument(request);
        System.out.println("The status of the document is " +
response.documentDescription().status());

    } catch (DocumentAlreadyExistsException e) {
        System.err.println("The document already exists. Moving on." );
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obtener más información sobre la API, consulta [CreateDocument](#) la Referencia AWS SDK for Java 2.x de la API.

CreateMaintenanceWindow

En el siguiente ejemplo de código, se muestra cómo usar CreateMaintenanceWindow.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static String createMaintenanceWindow(SsmClient ssmClient, String
winName) {
    CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
        .name(winName)
        .description("This is my maintenance window")
        .allowUnassociatedTargets(true)
        .duration(2)
        .cutoff(1)
        .schedule("cron(0 10 ? * MON-FRI *)")
        .build();

    try {
```



```
        CreateMaintenanceWindowResponse response =
ssmClient.createMaintenanceWindow(request);
        String maintenanceWindowId = response.windowId();
        System.out.println("The maintenance window id is " +
maintenanceWindowId);
        return maintenanceWindowId;

    } catch (DocumentAlreadyExistsException e) {
        System.err.println("The maintenance window already exists. Moving on.");
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
        .key("name")
        .values(winName)
        .build();

    DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
        .filters(filter)
        .build();

    String windowId = "";
    DescribeMaintenanceWindowsResponse response =
ssmClient.describeMaintenanceWindows(winRequest);
    List<MaintenanceWindowIdentity> windows = response.windowIdentities();
    if (!windows.isEmpty()) {
        windowId = windows.get(0).windowId();
        System.out.println("Window ID: " + windowId);
    } else {
        System.out.println("Window not found.");
    }
    return windowId;
}
```

- Para obtener más información sobre la API, consulta [CreateMaintenanceWindow](#) la Referencia AWS SDK for Java 2.x de la API.

CreateOpsItem

En el siguiente ejemplo de código, se muestra cómo usar CreateOpsItem.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Create an SSM OpsItem
public static String createSSMOpsItem(SsmClient ssmClient, String title, String
source, String category, String severity) {
    try {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
            .description("Created by the Systems Manager 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 obtener más información sobre la API, consulta [CreateOpsItem](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteDocument

En el siguiente ejemplo de código, se muestra cómo usar DeleteDocument.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Deletes an AWS Systems Manager document.
public static void deleteDoc(SsmClient ssmClient, String documentName) {
    try {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
            .name(documentName)
            .build();

        ssmClient.deleteDocument(documentRequest);
        System.out.println("The Systems Manager document was successfully
deleted.");

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

- Para obtener más información sobre la API, consulta [DeleteDocument](#) la Referencia AWS SDK for Java 2.x de la API.

DeleteMaintenanceWindow

En el siguiente ejemplo de código, se muestra cómo usar DeleteMaintenanceWindow.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void deleteMaintenanceWindow(SsmClient ssmClient, String winId) {
    try {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
            .windowId(winId)
            .build();

        ssmClient.deleteMaintenanceWindow(windowRequest);
        System.out.println("The maintenance window was successfully deleted.");


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

- Para obtener más información sobre la API, consulta [DeleteMaintenanceWindow](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeOpsItems

En el siguiente ejemplo de código, se muestra cómo usar DescribeOpsItems.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void describeOpsItems(SsmClient ssmClient, String key) {
    try {
        OpsItemFilter filter = OpsItemFilter.builder()
            .key(OpsItemFilterKey.OPS_ITEM_ID)
            .values(key)
            .operator(OpsItemFilterOperator.EQUAL)
            .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
            .maxResults(10)
            .opsItemFilters(filter)
            .build();

        DescribeOpsItemsResponse itemsResponse =
            ssmClient.describeOpsItems(itemsRequest);
        List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
        for (OpsItemSummary item : items) {
            System.out.println("The item title is " + item.title() + " and the
            status is "+item.status().toString());
        }

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

- Para obtener más información sobre la API, consulta [DescribeOpsItems](#) la Referencia AWS SDK for Java 2.x de la API.

DescribeParameters

En el siguiente ejemplo de código, se muestra cómo usar DescribeParameters.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [DescribeParameters](#) la Referencia AWS SDK for Java 2.x de la API.

PutParameter

En el siguiente ejemplo de código, se muestra cómo usar PutParameter.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [PutParameter](#) la Referencia AWS SDK for Java 2.x de la API.

SendCommand

En el siguiente ejemplo de código, se muestra cómo usar SendCommand.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Sends a SSM command to a managed node.
public static String sendSSMCommand(SsmClient ssmClient, String documentName,
String instanceId) throws InterruptedException {
    // Before we use Document to send a command - make sure it is active.
    boolean isDocumentActive = false;
    DescribeDocumentRequest request = DescribeDocumentRequest.builder()
        .name(documentName)
        .build();

    while (!isDocumentActive) {
        DescribeDocumentResponse response = ssmClient.describeDocument(request);
        String documentStatus = response.document().statusAsString();
        if (documentStatus.equals("Active")) {
            System.out.println("The Systems Manager document is active and ready
to use.");
            isDocumentActive = true;
        } else {
            System.out.println("The Systems Manager document is not active.
Status: " + documentStatus);
            try {
                // Add a delay to avoid making too many requests.
                Thread.sleep(5000); // Wait for 5 seconds before checking again
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
    }  
  }  
}  
  
// Create the SendCommandRequest.  
SendCommandRequest commandRequest = SendCommandRequest.builder()  
    .documentName(documentName)  
    .instanceIds(instanceId)  
    .build();  
  
// Send the command.  
SendCommandResponse commandResponse = ssmClient.sendCommand(commandRequest);  
String commandId = commandResponse.command().commandId();  
System.out.println("The command Id is " + commandId);  
  
// Wait for the command execution to complete.  
GetCommandInvocationRequest invocationRequest =  
GetCommandInvocationRequest.builder()  
    .commandId(commandId)  
    .instanceId(instanceId)  
    .build();  
  
System.out.println("Wait 5 secs");  
TimeUnit.SECONDS.sleep(5);  
  
// Retrieve the command execution details.  
GetCommandInvocationResponse commandInvocationResponse =  
ssmClient.getCommandInvocation(invocationRequest);  
  
// Check the status of the command execution.  
CommandInvocationStatus status = commandInvocationResponse.status();  
if (status == CommandInvocationStatus.SUCCESS) {  
    System.out.println("Command execution successful.");  
} else {  
    System.out.println("Command execution failed. Status: " + status);  
}  
return commandId;  
}
```

- Para obtener más información sobre la API, consulta [SendCommand](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateMaintenanceWindow

En el siguiente ejemplo de código, se muestra cómo usar UpdateMaintenanceWindow.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Update the maintenance window schedule
public static void updateSSMMaintenanceWindow(SsmClient ssmClient, String id,
String name) {
    try {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
            .windowId(id)
            .allowUnassociatedTargets(true)
            .duration(24)
            .enabled(true)
            .name(name)
            .schedule("cron(0 0 ? * MON *)")
            .build();

        ssmClient.updateMaintenanceWindow(updateRequest);
        System.out.println("The Systems Manager maintenance window was
successfully updated.");

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

- Para obtener más información sobre la API, consulta [UpdateMaintenanceWindow](#) la Referencia AWS SDK for Java 2.x de la API.

UpdateOpsItem

En el siguiente ejemplo de código, se muestra cómo usar UpdateOpsItem.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
public static void resolveOpsItem(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 obtener más información sobre la API, consulta [UpdateOpsItem](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Introducción a Systems Manager

El siguiente ejemplo de código muestra cómo trabajar con las ventanas, documentos y ventanas de mantenimiento de Systems Manager OpsItems.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.CommandInvocation;
import software.amazon.awssdk.services.ssm.model.CommandInvocationStatus;
import software.amazon.awssdk.services.ssm.model.CreateDocumentRequest;
import software.amazon.awssdk.services.ssm.model.CreateDocumentResponse;
import software.amazon.awssdk.services.ssm.model.CreateMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.CreateMaintenanceWindowResponse;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.DeleteDocumentRequest;
import software.amazon.awssdk.services.ssm.model.DeleteMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.DeleteOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.DescribeDocumentRequest;
import software.amazon.awssdk.services.ssm.model.DescribeDocumentResponse;
import software.amazon.awssdk.services.ssm.model.DescribeMaintenanceWindowsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeMaintenanceWindowsResponse;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsResponse;
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.DocumentType;
import software.amazon.awssdk.services.ssm.model.GetCommandInvocationRequest;
import software.amazon.awssdk.services.ssm.model.GetCommandInvocationResponse;
import software.amazon.awssdk.services.ssm.model.GetOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.GetOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.ListCommandInvocationsRequest;
import software.amazon.awssdk.services.ssm.model.ListCommandInvocationsResponse;
import software.amazon.awssdk.services.ssm.model.MaintenanceWindowFilter;
import software.amazon.awssdk.services.ssm.model.MaintenanceWindowIdentity;
import software.amazon.awssdk.services.ssm.model.OpsItemDataValue;
import software.amazon.awssdk.services.ssm.model.OpsItemFilter;
import software.amazon.awssdk.services.ssm.model.OpsItemFilterKey;
import software.amazon.awssdk.services.ssm.model.OpsItemFilterOperator;
import software.amazon.awssdk.services.ssm.model.OpsItemStatus;
```

```
import software.amazon.awssdk.services.ssm.model.OpsItemSummary;
import software.amazon.awssdk.services.ssm.model.SendCommandRequest;
import software.amazon.awssdk.services.ssm.model.SendCommandResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;
import software.amazon.awssdk.services.ssm.model.UpdateMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.UpdateOpsItemRequest;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html
 *
 * This Java program performs these tasks:
 * 1. Creates an AWS Systems Manager maintenance window with a default name or a
 * user-provided name.
 * 2. Modifies the maintenance window schedule.
 * 3. Creates a Systems Manager document with a default name or a user-provided
 * name.
 * 4. Sends a command to a specified EC2 instance using the created Systems Manager
 * document and displays the time when the command was invoked.
 * 5. Creates a Systems Manager OpsItem with a predefined title, source, category,
 * and severity.
 * 6. Updates and resolves the created OpsItem.
 * 7. Deletes the Systems Manager maintenance window, OpsItem, and document.
 */

public class SSMSscenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {
        String usage = ""
            Usage:
            <instanceId> <title> <source> <category> <severity>
    }
}
```

```

        Where:
            instanceId - The Amazon EC2 Linux/UNIX instance Id that AWS Systems
Manager uses (ie, i-0149338494ed95f06).
            title - The title of the parameter (default is Disk Space Alert).
            source - The source of the parameter (default is EC2).
            category - The category of the parameter. Valid values are
'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
Performance).
            severity - The severity of the parameter. Severity should be a
number from 1 to 4 (default is 2).
        """;

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

    Scanner scanner = new Scanner(System.in);
    String documentName;
    String windowName;
    String instanceId = args[0];
    String title = "Disk Space Alert" ;
    String source = "EC2" ;
    String category = "Performance" ;
    String severity = "2" ;

    Region region = Region.US_EAST_1;
    SsmClient ssmClient = SsmClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("""
        Welcome to the AWS Systems Manager SDK Getting Started scenario.
        This program demonstrates how to interact with Systems Manager using the
AWS SDK for Java (v2).
        Systems Manager is the operations hub for your AWS applications and
resources and a secure end-to-end management solution.
        The program's primary functions include creating a maintenance window,
creating a document, sending a command to a document,
        listing documents, listing commands, creating an OpsItem, modifying an
OpsItem, and deleting Systems Manager resources.
        Upon completion of the program, all AWS resources are cleaned up.
        Let's get started...
    """);

```

```
        Please hit Enter
        "");
    scanner.nextLine();
    System.out.println(DASHES);

    System.out.println("Create a Systems Manager maintenance window.");
    System.out.println("Please enter the maintenance window name (default is
    ssm-maintenance-window):");
    String win = scanner.nextLine();
    windowName = win.isEmpty() ? "ssm-maintenance-window" : win;
    String winId = createMaintenanceWindow(ssmClient, windowName);
    System.out.println(DASHES);

    System.out.println("Modify the maintenance window by changing the
    schedule");
    System.out.println("Please hit Enter");
    scanner.nextLine();
    updateSSMMaintenanceWindow(ssmClient, winId, windowName);
    System.out.println(DASHES);

    System.out.println("Create a document that defines the actions that Systems
    Manager performs on your EC2 instance.");
    System.out.println("Please enter the document name (default is
    ssmdocument):");
    String doc = scanner.nextLine();
    documentName = doc.isEmpty() ? "ssmdocument" : doc;
    createSSMDoc(ssmClient, documentName);

    System.out.println("Now we are going to run a command on an EC2 instance
    that echoes 'Hello, world!'");
    System.out.println("Please hit Enter");
    scanner.nextLine();
    String commandId = sendSSMCommand(ssmClient, documentName, instanceId);
    System.out.println(DASHES);

    System.out.println("Lets get the time when the specific command was sent to
    the specific managed node");
    System.out.println("Please hit Enter");
    scanner.nextLine();
    displayCommands(ssmClient, commandId);
    System.out.println(DASHES);

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


Now we will create a `Systems Manager OpsItem`.

An `OpsItem` is a feature provided by the `Systems Manager` service.

It is a type of operational data item that allows you to manage and track various operational issues, events, or tasks within your AWS environment.

You can create `OpsItems` to track and manage operational issues as they arise.

For example, you could create an `OpsItem` whenever your application detects a critical error

or an anomaly in your infrastructure.

```
""");
```

```
System.out.println("Please hit Enter");
scanner.nextLine();
String opsItemId = createSSMOpsItem(ssmClient, title, source, category,
severity);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Now we will update the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
String description = "An update to "+opsItemId ;
updateOpsItem(ssmClient, opsItemId, title, description);
System.out.println("Now we will get the status of the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
describeOpsItems(ssmClient, opsItemId);
System.out.println("Now we will resolve the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
resolveOpsItem(ssmClient, opsItemId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Would you like to delete the Systems Manager resources?
(y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    System.out.println("You selected to delete the resources.");
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    deleteOpsItem(ssmClient, opsItemId);
```

```
        deleteMaintenanceWindow(ssmClient, winId);
        deleteDoc(ssmClient, documentName);
    } else {
        System.out.println("The Systems Manager resources will not be deleted");
    }
    System.out.println(DASHES);

    System.out.println("This concludes the Systems Manager SDK Getting Started
scenario.");
    System.out.println(DASHES);
}

// Displays the date and time when the specific command was invoked.
public static void displayCommands(SsmClient ssmClient, String commandId) {
    try {
        ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
            .commandId(commandId)
            .build();

        ListCommandInvocationsResponse response =
ssmClient.listCommandInvocations(commandInvocationsRequest);
        List<CommandInvocation> commandList = response.commandInvocations();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
        for (CommandInvocation invocation : commandList) {
            System.out.println("The time of the command invocation is " +
formatter.format(invocation.requestedDateTime()));
        }

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

// Create an SSM OpsItem
public static String createSSMOpsItem(SsmClient ssmClient, String title, String
source, String category, String severity) {
    try {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
            .description("Created by the Systems Manager 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 "";
}

// Update the AWS SSM OpsItem.
public static void updateOpsItem(SsmClient ssmClient, String opsItemId, String
title, String description) {
    Map<String, OpsItemDataValue> operationalData = new HashMap<>();
    operationalData.put("key1",
OpsItemDataValue.builder().value("value1").build());
    operationalData.put("key2",
OpsItemDataValue.builder().value("value2").build());

    try {
        UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
            .opsItemId(opsItemId)
            .title(title)
            .operationalData(operationalData)
            .status(getOpsItem(ssmClient, opsItemId))
            .description(description)
            .build();

        ssmClient.updateOpsItem(request);

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

public static void resolveOpsItem(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);
}
}

// Gets a specific OpsItem.
private static OpsItemStatus getOpsItem(SsmClient ssmClient, String opsItemId) {
    GetOpsItemRequest itemRequest = GetOpsItemRequest.builder()
        .opsItemId(opsItemId)
        .build();

    try {
        GetOpsItemResponse response = ssmClient.getOpsItem(itemRequest);
        return response.opsItem().status();
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}

// Sends a SSM command to a managed node.
public static String sendSSMCommand(SsmClient ssmClient, String documentName,
String instanceId) throws InterruptedException {
    // Before we use Document to send a command - make sure it is active.
    boolean isDocumentActive = false;
    DescribeDocumentRequest request = DescribeDocumentRequest.builder()
        .name(documentName)
        .build();

    while (!isDocumentActive) {
        DescribeDocumentResponse response = ssmClient.describeDocument(request);
        String documentStatus = response.document().statusAsString();
        if (documentStatus.equals("Active")) {
            System.out.println("The Systems Manager document is active and ready
to use.");
        }
    }
}
```

```
        isDocumentActive = true;
    } else {
        System.out.println("The Systems Manager document is not active.
Status: " + documentStatus);
        try {
            // Add a delay to avoid making too many requests.
            Thread.sleep(5000); // Wait for 5 seconds before checking again
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

// Create the SendCommandRequest.
SendCommandRequest commandRequest = SendCommandRequest.builder()
    .documentName(documentName)
    .instanceIds(instanceId)
    .build();

// Send the command.
SendCommandResponse commandResponse = ssmClient.sendCommand(commandRequest);
String commandId = commandResponse.command().commandId();
System.out.println("The command Id is " + commandId);

// Wait for the command execution to complete.
GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
    .commandId(commandId)
    .instanceId(instanceId)
    .build();

System.out.println("Wait 5 secs");
TimeUnit.SECONDS.sleep(5);

// Retrieve the command execution details.
GetCommandInvocationResponse commandInvocationResponse =
ssmClient.getCommandInvocation(invocationRequest);

// Check the status of the command execution.
CommandInvocationStatus status = commandInvocationResponse.status();
if (status == CommandInvocationStatus.SUCCESS) {
    System.out.println("Command execution successful.");
} else {
    System.out.println("Command execution failed. Status: " + status);
}
```

```
    }
    return commandId;
}

// Deletes an AWS Systems Manager document.
public static void deleteDoc(SsmClient ssmClient, String documentName) {
    try {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
            .name(documentName)
            .build();

        ssmClient.deleteDocument(documentRequest);
        System.out.println("The Systems Manager document was successfully
deleted.");

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

public static void deleteMaintenanceWindow(SsmClient ssmClient, String winId) {
    try {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
            .windowId(winId)
            .build();

        ssmClient.deleteMaintenanceWindow(windowRequest);
        System.out.println("The maintenance window was successfully deleted.");

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

// Update the maintenance window schedule
public static void updateSSMMaintenanceWindow(SsmClient ssmClient, String id,
String name) {
    try {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
            .windowId(id)
```

```
        .allowUnassociatedTargets(true)
        .duration(24)
        .enabled(true)
        .name(name)
        .schedule("cron(0 0 ? * MON *)")
        .build();

    ssmClient.updateMaintenanceWindow(updateRequest);
    System.out.println("The Systems Manager maintenance window was
successfully updated.");

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

public static String createMaintenanceWindow(SsmClient ssmClient, String
winName) {
    CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
        .name(winName)
        .description("This is my maintenance window")
        .allowUnassociatedTargets(true)
        .duration(2)
        .cutoff(1)
        .schedule("cron(0 10 ? * MON-FRI *)")
        .build();

    try {
        CreateMaintenanceWindowResponse response =
ssmClient.createMaintenanceWindow(request);
        String maintenanceWindowId = response.windowId();
        System.out.println("The maintenance window id is " +
maintenanceWindowId);
        return maintenanceWindowId;

    } catch (DocumentAlreadyExistsException e) {
        System.err.println("The maintenance window already exists. Moving on.");
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
    .key("name")
    .values(winName)
    .build();

DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
    .filters(filter)
    .build();

String windowId = "";
DescribeMaintenanceWindowsResponse response =
ssmClient.describeMaintenanceWindows(winRequest);
List<MaintenanceWindowIdentity> windows = response.windowIdentities();
if (!windows.isEmpty()) {
    windowId = windows.get(0).windowId();
    System.out.println("Window ID: " + windowId);
} else {
    System.out.println("Window not found.");
}
return windowId;
}

// Create an AWS SSM document to use in this scenario.
public static void createSSMDoc(SsmClient ssmClient, String docName) {
    // Create JSON for the content
    String jsonData = ""
    {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                    "runCommand": [
                        "echo 'Hello, world!'"
                    ]
                }
            }
        ]
    }
    "";
```



```
try {
    CreateDocumentRequest request = CreateDocumentRequest.builder()
        .content(jsonData)
        .name(docName)
        .documentType(DocumentType.COMMAND)
        .build();

    // Create the document.
    CreateDocumentResponse response = ssmClient.createDocument(request);
    System.out.println("The status of the document is " +
response.documentDescription().status());

} catch (DocumentAlreadyExistsException e) {
    System.err.println("The document already exists. Moving on." );
} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void describeOpsItems(SsmClient ssmClient, String key) {
    try {
        OpsItemFilter filter = OpsItemFilter.builder()
            .key(OpsItemFilterKey.OPS_ITEM_ID)
            .values(key)
            .operator(OpsItemFilterOperator.EQUAL)
            .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
            .maxResults(10)
            .opsItemFilters(filter)
            .build();

        DescribeOpsItemsResponse itemsResponse =
ssmClient.describeOpsItems(itemsRequest);
        List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
        for (OpsItemSummary item : items) {
            System.out.println("The item title is " + item.title() +" and the
status is "+item.status().toString());
        }

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

```
    }  
  }  
  
  public static void deleteOpsItem(SsmClient ssmClient, String opsId) {  
    try {  
      DeleteOpsItemRequest deleteOpsItemRequest =  
DeleteOpsItemRequest.builder()  
        .opsItemId(opsId)  
        .build();  
  
      ssmClient.deleteOpsItem(deleteOpsItemRequest);  
      System.out.println(opsId + " Opsitem was deleted");  
  
    } catch (SsmException e) {  
      System.err.println(e.getMessage());  
      System.exit(1);  
    }  
  }  
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
 - [CommandInvocations](#)
 - [CreateDocument](#)
 - [CreateMaintenanceWindow](#)
 - [CreateOpsItem](#)
 - [DeleteMaintenanceWindow](#)
 - [SendCommand](#)
 - [UpdateOpsItem](#)

Ejemplos de Amazon Textract usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x con Amazon Textract.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)

Acciones

AnalyzeDocument

En el siguiente ejemplo de código, se muestra cómo usar AnalyzeDocument.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [AnalyzeDocument](#) la Referencia AWS SDK for Java 2.x de la API.

DetectDocumentText

En el siguiente ejemplo de código, se muestra cómo usar DetectDocumentText.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Detectar texto de un 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 un documento ubicado en un bucket de 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 =
textractClient.detectDocumentText(detectDocumentTextRequest);
        for (Block block : textResponse.blocks()) {
            System.out.println("The block type is " +
block.blockType().toString());
        }
    }
}
```

```
    }

    DocumentMetadata documentMetadata = textResponse.documentMetadata();
    System.out.println("The number of pages in the document is " +
documentMetadata.pages());

    } catch (TextractException e) {

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

- Para obtener más información sobre la API, consulta [DetectDocumentText](#) la Referencia AWS SDK for Java 2.x de la API.

StartDocumentAnalysis

En el siguiente ejemplo de código, se muestra cómo usar StartDocumentAnalysis.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [StartDocumentAnalysis](#) la Referencia AWS SDK for Java 2.x de la API.

Ejemplos de Amazon Transcribe usando SDK para Java 2.x

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Java 2.x con Amazon Transcribe.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

ListTranscriptionJobs

En el siguiente ejemplo de código, se muestra cómo usar `ListTranscriptionJobs`.

SDK para Java 2.x

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [ListTranscriptionJobs](#) la Referencia AWS SDK for Java 2.x de la API.

StartTranscriptionJob

En el siguiente ejemplo de código, se muestra cómo usar StartTranscriptionJob.

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 obtener más información sobre la API, consulta [StartTranscriptionJob](#) la Referencia AWS SDK for Java 2.x de la API.

Escenarios

Transcribir audio y obtener datos de trabajo

En el siguiente ejemplo de código, se muestra cómo:

- Iniciar un trabajo de transcripción con Amazon Transcribe.
- Esperar a que el trabajo finalice.
- Obtener el URI en el que está almacenada la transcripción.

Para obtener información, consulte [Introducción a Amazon Transcribe](#).

SDK para Java 2.x

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Transcribe un archivo 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();
}
}
```

Transcribe el audio en streaming desde el micrófono del equipo.

```
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 obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
 - [GetTranscriptionJob](#)
 - [StartTranscriptionJob](#)

Ejemplos de servicios cruzados con SDK para Java 2.x

En las siguientes aplicaciones de ejemplo se utiliza AWS SDK for Java 2.x para funcionar en varios Servicios de AWS.

Los ejemplos de servicios combinados apuntan a un nivel avanzado de experiencia para ayudarle a empezar a crear aplicaciones.

Ejemplos

- [Creación de una aplicación para enviar datos a una tabla de DynamoDB](#)
- [Crear un chatbot de Amazon Lex para atraer visitantes a su sitio web](#)
- [Creación de una aplicación de publicación y suscripción que traduzca mensajes](#)
- [Crear una aplicación web que envíe y recupere mensajes mediante Amazon SQS](#)
- [Creación de una aplicación de administración de activos fotográficos que permita a los usuarios administrar las fotos mediante etiquetas](#)
- [Creación de una aplicación web para hacer un seguimiento de los datos de DynamoDB](#)
- [Crear un rastreador de artículos de Amazon Redshift](#)
- [Crear un rastreador de elementos de trabajo de Aurora Serverless](#)
- [Creación de una aplicación que analice los comentarios de los clientes y sintetice el audio](#)
- [Detecte el PPE en las imágenes con Amazon Rekognition AWS mediante un SDK](#)
- [Detecte objetos en imágenes con Amazon Rekognition AWS mediante un SDK](#)
- [Detecte personas y objetos en un vídeo con Amazon Rekognition AWS mediante un SDK](#)
- [Supervise el rendimiento de Amazon DynamoDB mediante un SDK AWS](#)
- [Uso de API Gateway para invocar una función de Lambda](#)
- [Uso de Step Functions para invocar funciones de Lambda](#)
- [Usar eventos programados para invocar una función de Lambda](#)

Creación de una aplicación para enviar datos a una tabla de DynamoDB

SDK para Java 2.x

Indica cómo crear una aplicación web dinámica que envíe datos mediante la API Java de Amazon DynamoDB y un mensaje de texto mediante la API Java de Amazon Simple Notification Service.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- DynamoDB
- Amazon SNS

Crear un chatbot de Amazon Lex para atraer visitantes a su sitio web

SDK para Java 2.x

Indica cómo utilizar la API de Amazon Lex para crear un Chatbot dentro de una aplicación web con el fin de atraer visitantes a su sitio web.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Creación de una aplicación de publicación y suscripción que traduzca mensajes

SDK para Java 2.x

Indica cómo utilizar la API de Java de Amazon Simple Notification Service para crear una aplicación web con funcionalidad de suscripción y publicación. Además, esta aplicación de ejemplo también traduce los mensajes.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#).

Para obtener el código fuente completo y las instrucciones sobre cómo configurar y ejecutar el ejemplo que usa la API Java Async, consulta el ejemplo completo en [GitHub](#)

Servicios utilizados en este ejemplo

- Amazon SNS
- Amazon Translate

Crear una aplicación web que envíe y recupere mensajes mediante Amazon SQS

SDK para Java 2.x

Muestra cómo usar la API de Amazon SQS para desarrollar una API de REST de Spring que envíe y recupere mensajes.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#)

Servicios utilizados en este ejemplo

- Amazon Comprehend
- Amazon SQS

Creación de una aplicación de administración de activos fotográficos que permita a los usuarios administrar las fotos mediante etiquetas

SDK para Java 2.x

Muestra cómo desarrollar una aplicación de gestión de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Creación de una aplicación web para hacer un seguimiento de los datos de DynamoDB

SDK para Java 2.x

Muestra cómo utilizar la API de Amazon DynamoDB para crear una aplicación web dinámica que haga un seguimiento de los datos de trabajo de DynamoDB.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- DynamoDB
- Amazon SES

Crear un rastreador de artículos de Amazon Redshift

SDK para Java 2.x

Muestra cómo crear una aplicación web que realice un seguimiento de los elementos de trabajo almacenados en una base de datos de Amazon Redshift e informe al respecto.

Para obtener el código fuente completo y las instrucciones sobre cómo configurar una API REST de Spring que consulte los datos de Amazon Redshift y para que la utilice una aplicación de React, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Redshift
- Amazon SES

Crear un rastreador de elementos de trabajo de Aurora Serverless

SDK para Java 2.x

Muestra cómo crear una aplicación web que haga un seguimiento de los elementos de trabajo almacenados en una base de datos de Amazon RDS e informe al respecto.

Para obtener el código fuente completo y las instrucciones sobre cómo configurar una API REST de Spring que consulte los datos de Amazon Aurora Serverless y para que la utilice una aplicación React, consulte el ejemplo completo en [GitHub](#).

Para obtener el código fuente completo y las instrucciones sobre cómo configurar y ejecutar un ejemplo que utilice la API JDBC, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

Creación de una aplicación que analice los comentarios de los clientes y sintetice el audio

SDK para Java 2.x

Esta aplicación de ejemplo analiza y almacena las tarjetas de comentarios de los clientes. Concretamente, satisface la necesidad de un hotel ficticio en la ciudad de Nueva York. El hotel recibe comentarios de los huéspedes en varios idiomas en forma de tarjetas de comentarios físicas. Esos comentarios se cargan en la aplicación a través de un cliente web. Una vez cargada la imagen de una tarjeta de comentarios, se llevan a cabo los siguientes pasos:

- El texto se extrae de la imagen mediante Amazon Textract.
- Amazon Comprehend determina la opinión del texto extraído y su idioma.
- El texto extraído se traduce al inglés mediante Amazon Translate.
- Amazon Polly sintetiza un archivo de audio a partir del texto extraído.

La aplicación completa se puede implementar con AWS CDK. Para ver el código fuente y las instrucciones de implementación, consulta el proyecto en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Detecte el PPE en las imágenes con Amazon Rekognition AWS mediante un SDK

SDK para Java 2.x

Muestra cómo crear una AWS Lambda función que detecte imágenes con un equipo de protección personal.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Detecte objetos en imágenes con Amazon Rekognition AWS mediante un SDK

SDK para Java 2.x

Muestra cómo utilizar la API de Java de Amazon Rekognition para crear una aplicación que utilice Amazon Rekognition para identificar objetos por categoría en imágenes ubicadas en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en [GitHub](#)

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detecte personas y objetos en un vídeo con Amazon Rekognition AWS mediante un SDK

SDK para Java 2.x

Muestra cómo utilizar la API Java de Amazon Rekognition para crear una aplicación que detecte rostros y objetos en vídeos ubicados en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en [GitHub](#)

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Supervise el rendimiento de Amazon DynamoDB mediante un SDK AWS

SDK para Java 2.x

En este ejemplo se muestra cómo configurar una aplicación Java para supervisar el rendimiento de DynamoDB. La aplicación envía los datos métricos a los CloudWatch que puede supervisar el rendimiento.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- CloudWatch
- DynamoDB

Uso de API Gateway para invocar una función de Lambda

SDK para Java 2.x

Muestra cómo crear una AWS Lambda función mediante la API de tiempo de ejecución Lambda Java. En este ejemplo, se invocan diferentes AWS servicios para realizar un caso de uso específico. En este ejemplo se indica cómo crear una función de Lambda invocada por Amazon API Gateway que escanea una tabla de Amazon DynamoDB en busca de aniversarios laborales y utiliza Amazon Simple Notification Service (Amazon SNS) para enviar un mensaje de texto a sus empleados que les felicite en la fecha de su primer aniversario.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Uso de Step Functions para invocar funciones de Lambda

SDK para Java 2.x

Muestra cómo crear un flujo de trabajo AWS sin servidor mediante AWS Step Functions y el AWS SDK for Java 2.x. Cada paso del flujo de trabajo se implementa mediante una AWS Lambda función.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- DynamoDB

- Lambda
- Amazon SES
- Step Functions

Usar eventos programados para invocar una función de Lambda

SDK para Java 2.x

Muestra cómo crear un evento EventBridge programado de Amazon que invoque una AWS Lambda función. Configure EventBridge para usar una expresión cron para programar cuándo se invoca la función Lambda. En este ejemplo, creará una función de Lambda utilizando la API de tiempo de ejecución de Lambda Java. En este ejemplo, se invocan diferentes AWS servicios para realizar un caso de uso específico. Este ejemplo indica cómo crear una aplicación que envíe un mensaje de texto a sus empleados para felicitarles por su primer aniversario.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Seguridad para AWS SDK for Java

La seguridad en la nube de Amazon Web Services (AWS) es la máxima prioridad. Como cliente de AWS, se beneficia de una arquitectura de red y un centro de datos que se han diseñado para satisfacer los requisitos de seguridad de las organizaciones más exigentes. La seguridad es una responsabilidad compartida entre AWS usted y usted. En el [modelo de responsabilidad compartida](#), se habla de “seguridad de la nube” y “seguridad en la nube”:

Seguridad de la nube: AWS se encarga de proteger la infraestructura en la que se ejecutan todos los servicios que se ofrecen en la AWS nube y de proporcionarle servicios que pueda utilizar de forma segura. Nuestra responsabilidad en materia de seguridad es nuestra máxima prioridad AWS, y auditores externos comprueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los [programas de AWS conformidad](#).

Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice y otros factores, como la confidencialidad de sus datos, los requisitos de su organización y las leyes y reglamentos aplicables.

Este AWS producto o servicio sigue el [modelo de responsabilidad compartida](#) a través de los servicios específicos de Amazon Web Services (AWS) a los que da soporte. Para obtener información sobre la seguridad de los AWS servicios, consulte la [página de documentación sobre la seguridad del AWS servicio](#) y [AWS los servicios que se encuentran dentro del ámbito de aplicación de AWS las medidas de conformidad establecidas por el programa de conformidad](#).

Temas

- [Protección de datos en la versión AWS SDK for Java 2.x](#)
- [Trabajar con TLS en el SDK para JavaScript](#)
- [Identity and Access Management](#)
- [Validación de la conformidad de este AWS producto o servicio](#)
- [Resiliencia de este AWS producto o servicio](#)
- [Seguridad de la infraestructura para este AWS producto o servicio](#)

Protección de datos en la versión AWS SDK for Java 2.x

El modelo de [responsabilidad AWS compartida modelo](#) se aplica a la protección de datos en AWS SDK for Java. Como se describe en este modelo, AWS es responsable de proteger la infraestructura

global que ejecuta todos los Nube de AWS. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Usted también es responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las [Preguntas frecuentes sobre la privacidad de datos](#). Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog sobre el [Modelo de responsabilidad compartida de AWS y GDPR](#) en el Blog de seguridad de AWS .

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utilice la autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos. AWS Se recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con. AWS CloudTrail
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utilice servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-2 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte [Estándar de procesamiento de la información federal \(FIPS\) 140-2](#).

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como, por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja con el SDK para Java u otro Servicios de AWS mediante la consola, la API o AWS los SDK. AWS CLI Cualquier dato que ingrese en etiquetas o campos de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Trabajar con TLS en el SDK para JavaScript

AWS SDK for Java Utiliza las capacidades de TLS de su plataforma Java subyacente. En este tema, mostramos ejemplos del uso de la implementación de OpenJDK utilizada por [Amazon Corretto 17](#).

Para que funcione Servicios de AWS, el JDK subyacente debe admitir una versión mínima de TLS 1.2, pero se recomienda TLS 1.3.

Los usuarios deben consultar la documentación de la plataforma Java que utilizan con el SDK para saber qué versiones de TLS están habilitadas de forma predeterminada y cómo habilitar y deshabilitar versiones específicas de TLS.

Cómo comprobar la información de la versión de TLS

Con OpenJDK, el código siguiente muestra el uso de [SSLContext](#) para imprimir qué versiones de TLS/SSL son compatibles.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()));
```

Por ejemplo, Amazon Corretto 17 (OpenJDK) produce el siguiente resultado.

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

Para ver el protocolo de enlace SSL en acción y qué versión de TLS se utiliza, puede utilizar la propiedad del sistema `javax.net.debug`.

Por ejemplo, ejecute una aplicación Java que utilice TLS.

```
java app.jar -Djavax.net.debug=ssl:handshake
```

La aplicación registra el enlace SSL de forma similar a la siguiente.

```
...
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 una versión mínima de TLS

El SDK para Java siempre prefiere la última versión de TLS compatible con la plataforma y el servicio. Si desea aplicar una versión mínima específica de TLS, consulte la documentación de su plataforma Java.

Para las JVM basadas en OpenJDK, puede utilizar la propiedad del sistema `jdk.tls.client.protocols`.

Por ejemplo, si desea que los clientes del servicio SDK de su aplicación utilicen TLS 1.2, aunque TLS 1.3 esté disponible, proporcione la siguiente propiedad del sistema.

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

AWS Los puntos finales de la API se actualizan a TLS 1.2

Consulte esta [entrada del blog](#) para obtener información sobre la migración de los puntos finales de la AWS API a TLS 1.2 para obtener la versión mínima.

Identity and Access Management

AWS Identity and Access Management (IAM) es una herramienta Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los AWS recursos. Los administradores de IAM controlan quién puede autenticarse (iniciar sesión) y quién puede autorizarse (tener permisos) para usar los recursos. AWS La IAM es una Servicio de AWS opción que puede utilizar sin coste adicional.

Temas

- [Público](#)
- [Autenticación con identidades](#)

- [Administración de acceso mediante políticas](#)
- [¿Cómo Servicios de AWS trabajar con IAM](#)
- [Solución de problemas de AWS identidad y acceso](#)

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo en el que se realice. AWS

Usuario del servicio: si Servicios de AWS solía hacer su trabajo, el administrador le proporcionará las credenciales y los permisos que necesita. A medida que vaya utilizando más AWS funciones para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una función de AWS, consulte [Solución de problemas de AWS identidad y acceso](#) o consulte la guía del usuario de la Servicio de AWS que está utilizando.

Administrador de servicios: si está a cargo de AWS los recursos de su empresa, probablemente tenga acceso total a ellos AWS. Su trabajo consiste en determinar a qué AWS funciones y recursos deben acceder los usuarios del servicio. Luego, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo su empresa puede utilizar la IAM AWS, consulte la guía del usuario del Servicio de AWS que está utilizando.

Administrador de IAM: si es un administrador de IAM, es posible que quiera conocer más detalles sobre cómo escribir políticas para administrar el acceso a AWS. Para ver ejemplos de políticas AWS basadas en la identidad que puede utilizar en IAM, consulte la guía del usuario de la Servicio de AWS que está utilizando.

Autenticación con identidades

La autenticación es la forma de iniciar sesión AWS con sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario de IAM o asumiendo una función de IAM. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (Centro de identidades de IAM), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su

administrador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, estás asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte [Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario](#).

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte [Firmar las solicitudes de la AWS API](#) en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte [Autenticación multifactor](#) en la Guía del usuario de AWS IAM Identity Center y [Uso de la autenticación multifactor \(MFA\) en AWS](#) en la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulte [Tareas que requieren credenciales de usuario raíz](#) en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidades web AWS Directory Service, el directorio del Centro de Identidad o cualquier usuario al que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de

identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utilice AWS IAM Identity Center. Puede crear usuarios y grupos en el Centro de identidades de IAM, o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus Cuentas de AWS aplicaciones. Para obtener más información, consulte [¿Qué es el Centro de identidades de IAM?](#) en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un [usuario de IAM](#) es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte [Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración](#) en la Guía del usuario de IAM.

Un [grupo de IAM](#) es una identidad que especifica un conjunto de usuarios de IAM. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales permanentes a largo plazo y los roles proporcionan credenciales temporales. Para más información, consulte [Cuándo crear un usuario de IAM \(en lugar de un rol\)](#) en la Guía del usuario de IAM.

Roles de IAM

Un [rol de IAM](#) es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente una función de IAM en el AWS Management Console [cambiando](#) de función. Puede asumir un rol llamando a una operación de AWS API AWS CLI o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte [Uso de roles de IAM](#) en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- **Acceso de usuario federado:** para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles para federación, consulte [Creación de un rol para un proveedor de identidades de terceros](#) en la Guía del usuario de IAM. Si utiliza IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulte [Conjuntos de permisos](#) en la Guía del usuario de AWS IAM Identity Center .
- **Permisos de usuario de IAM temporales:** un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- **Acceso entre cuentas:** puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunas Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para obtener información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte [Acceso a recursos entre cuentas en IAM](#) en la Guía del usuario de IAM.
- **Acceso entre servicios:** algunos Servicios de AWS utilizan funciones en otros Servicios de AWS. Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
- **Sesiones de acceso directo (FAS):** cuando utilizas un usuario o un rol de IAM para realizar acciones en ellas AWS, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte [Reenviar sesiones de acceso](#).
- **Rol de servicio:** un rol de servicio es un [rol de IAM](#) que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio

desde IAM. Para obtener más información, consulte [Creación de un rol para delegar permisos a un Servicio de AWS](#) en la Guía del usuario de IAM.

- **Función vinculada al servicio:** una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.
- **Aplicaciones que se ejecutan en Amazon EC2:** puede usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una instancia EC2 y realizan AWS CLI solicitudes a la API. AWS Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia de EC2. Para asignar una AWS función a una instancia EC2 y ponerla a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia de EC2 obtener credenciales temporales. Para más información, consulte [Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2](#) en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte [Cuándo crear un rol de IAM \(en lugar de un usuario\)](#) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte [Información general de políticas JSON](#) en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte [Creación de políticas de IAM](#) en la Guía del usuario de IAM.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para más información sobre cómo elegir una política administrada o una política insertada, consulte [Elegir entre políticas administradas y políticas insertadas](#) en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe [especificar una entidad principal](#) en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios que admiten las ACL. AWS WAF Para obtener más información sobre las ACL, consulte [Información general de Lista de control de acceso \(ACL\)](#) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- **Límites de permisos:** un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM (usuario o rol de IAM). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte [Límites de permisos para las entidades de IAM](#) en la Guía del usuario de IAM.
- **Políticas de control de servicios (SCP):** las SCP son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations es un servicio para agrupar y gestionar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilita todas las características en una organización, entonces podrá aplicar políticas de control de servicio (SCP) a una o a todas sus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una. Usuario raíz de la cuenta de AWS Para obtener más información acerca de Organizations y las SCP, consulte [Funcionamiento de las SCP](#) en la Guía del usuario de AWS Organizations .
- **Políticas de sesión:** las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte [Políticas de sesión](#) en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo AWS determinar si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la [lógica de evaluación de políticas](#) en la Guía del usuario de IAM.

¿Cómo Servicios de AWS trabajar con IAM

Para obtener una visión general de cómo Servicios de AWS trabajar con la mayoría de las funciones de IAM, consulte [AWS los servicios que funcionan con IAM en la Guía del usuario de IAM](#).

Para obtener información sobre cómo utilizar una función específica Servicio de AWS con IAM, consulte la sección de seguridad de la guía del usuario del servicio correspondiente.

Solución de problemas de AWS identidad y acceso

Utilice la siguiente información como ayuda para diagnosticar y solucionar los problemas habituales que pueden surgir al trabajar con un AWS IAM.

Temas

- [No estoy autorizado a realizar ninguna acción en AWS](#)
- [No estoy autorizado a realizar tareas como: PassRole](#)
- [Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis AWS recursos](#)

No estoy autorizado a realizar ninguna acción en AWS

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM mateojackson intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *my-example-widget*, pero no tiene los permisos ficticios *aws:GetWidget*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws:GetWidget on resource: my-example-widget
```

En este caso, la política del usuario mateojackson debe actualizarse para permitir el acceso al recurso *my-example-widget* mediante la acción *aws:GetWidget*.

Si necesita ayuda, póngase en contacto con su AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

No estoy autorizado a realizar tareas como: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, las políticas deben actualizarse a fin de permitirle pasar un rol a AWS.

Algunos Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en AWS. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su administrador. AWS El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis AWS recursos

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para más información, consulte lo siguiente:

- Para saber si AWS es compatible con estas funciones, consulte [¿Cómo Servicios de AWS trabajar con IAM.](#)

- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte [Proporcionar acceso a un usuario de IAM en otro usuario de su propiedad Cuenta de AWS en](#) la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo [proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros](#) en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte [Proporcionar acceso a usuarios autenticados externamente \(identidad federada\)](#) en la Guía del usuario de IAM.
- Para obtener información acerca del uso de roles y políticas basadas en recursos para el acceso entre cuentas, consulte [Acceso a recursos entre cuentas en IAM](#) en la Guía del usuario de IAM.

Validación de la conformidad de este AWS producto o servicio

Para saber si un programa de cumplimiento Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte [Servicios de AWS Alcance por programa](#) de de cumplimiento y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de [AWS cumplimiento > Programas AWS](#) .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte [Descarga de informes en AWS Artifact](#) .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- [Guías de inicio rápido sobre seguridad y cumplimiento](#): estas guías de implementación analizan las consideraciones arquitectónicas y proporcionan los pasos para implementar entornos básicos centrados en AWS la seguridad y el cumplimiento.
- Diseño de [arquitectura para garantizar la seguridad y el cumplimiento de la HIPAA en Amazon Web Services](#): en este documento técnico se describe cómo pueden utilizar AWS las empresas para crear aplicaciones aptas para la HIPAA.

Note

No Servicios de AWS todas cumplen los requisitos de la HIPAA. Para más información, consulte la [Referencia de servicios compatibles con HIPAA](#).

- [AWS Recursos de](#) de cumplimiento: esta colección de libros de trabajo y guías puede aplicarse a su industria y ubicación.
- [AWS Guías de cumplimiento para clientes](#): comprenda el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. Las guías resumen las mejores prácticas para garantizar la seguridad Servicios de AWS y orientan los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST), el Consejo de Normas de Seguridad del Sector de Tarjetas de Pago (PCI) y la Organización Internacional de Normalización (ISO)).
- [Evaluación de los recursos con reglas](#) en la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- [AWS Security Hub](#)— Esto Servicio de AWS proporciona una visión completa del estado de su seguridad interior AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulte la [Referencia de controles de Security Hub](#).
- [Amazon GuardDuty](#): Servicio de AWS detecta posibles amenazas para sus cargas de trabajo Cuentas de AWS, contenedores y datos mediante la supervisión de su entorno para detectar actividades sospechosas y maliciosas. GuardDuty puede ayudarlo a cumplir con varios requisitos de conformidad, como el PCI DSS, al cumplir con los requisitos de detección de intrusiones exigidos por ciertos marcos de cumplimiento.
- [AWS Audit Manager](#)— Esto le Servicio de AWS ayuda a auditar continuamente su AWS uso para simplificar la gestión del riesgo y el cumplimiento de las normativas y los estándares del sector.

Este AWS producto o servicio sigue el [modelo de responsabilidad compartida](#) a través de los servicios específicos de Amazon Web Services (AWS) a los que da soporte. Para obtener información sobre la seguridad de los AWS servicios, consulte la [página de documentación sobre la seguridad del AWS servicio](#) y [AWS los servicios que se encuentran dentro del ámbito de aplicación de AWS las medidas de conformidad establecidas por el programa de conformidad](#).

Resiliencia de este AWS producto o servicio

La infraestructura AWS global se basa en Regiones de AWS zonas de disponibilidad.

Regiones de AWS proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia.

Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

[Para obtener más información sobre AWS las regiones y las zonas de disponibilidad, consulte Infraestructura global.AWS](#)

Este AWS producto o servicio sigue el [modelo de responsabilidad compartida](#) a través de los servicios específicos de Amazon Web Services (AWS) a los que da soporte. Para obtener información sobre la seguridad de los AWS servicios, consulte la [página de documentación sobre la seguridad del AWS servicio](#) y [AWS los servicios que se encuentran dentro del ámbito de aplicación de AWS las medidas de conformidad establecidas por el programa de conformidad](#).

Seguridad de la infraestructura para este AWS producto o servicio

Este AWS producto o servicio utiliza servicios gestionados y, por lo tanto, está protegido por la seguridad de la red AWS global. Para obtener información sobre los servicios AWS de seguridad y cómo se AWS protege la infraestructura, consulte [Seguridad AWS en la nube](#). Para diseñar su AWS entorno utilizando las mejores prácticas de seguridad de la infraestructura, consulte [Protección de infraestructuras en un marco](#) de buena AWS arquitectura basado en el pilar de la seguridad.

Utiliza las llamadas a la API AWS publicadas para acceder a este AWS producto o servicio a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad de seguridad de IAM principal. También puede

utilizar [AWS Security Token Service](#) (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Este AWS producto o servicio sigue el [modelo de responsabilidad compartida](#) a través de los servicios específicos de Amazon Web Services (AWS) a los que da soporte. Para obtener información sobre la seguridad de los AWS servicios, consulte la [página de documentación sobre la seguridad del AWS servicio](#) y [AWS los servicios que se encuentran dentro del ámbito de aplicación de AWS las medidas de conformidad establecidas por el programa de conformidad](#).

Migre de la versión 1.x a la 2.x del AWS SDK for Java

La AWS SDK for Java 2.x es una reescritura importante del código base 1.x creado sobre Java 8+. Incluye muchas actualizaciones como, por ejemplo, coherencia mejorada, facilidad de uso e inmutabilidad sólidamente aplicada. En esta sección se describen las principales características nuevas de la versión 2.x y se explica cómo migrar el código de la versión 1.x a la 2.x.

Temas

- [¿Qué novedades incluye la versión 2?](#)
- [step-by-step Instrucciones de migración con ejemplo](#)
- [Nombre del paquete para Maven ArtifactID mappings](#)
- [Qué diferencia hay entre la AWS SDK for Java 1.x y la 2.x](#)
- [Utilizar el SDK para Java 1.x y 2.x en paralelo](#)

¿Qué novedades incluye la versión 2?

- Puede configurar sus propios clientes HTTP. Consulte [Configuración del transporte HTTP](#).
- Los clientes asíncronos ofrecen soporte de E/S sin bloqueo y devuelven objetos. `CompletableFuture` Consulte [Programación asíncrona](#).
- Las operaciones que devuelven varias páginas tienen respuestas paginadas automáticamente. De esta forma, puede centrar su código en qué hacer con la respuesta, sin necesidad de comprobar y obtener páginas posteriores. Consulte [Paginación](#).
- Se ha mejorado el rendimiento de las funciones a la hora de inicio del SDK. AWS Lambda Consulte [Mejoras en el rendimiento del tiempo de inicio del SDK](#).
- La versión 2.x admite un nuevo método abreviado para crear solicitudes.

Example

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

Para obtener más detalles sobre las nuevas características y ver ejemplos de código específicos, consulte las demás secciones de esta guía.

- [Quick Start \(Inicio rápido\)](#)

- [Configuración](#)
- [Ejemplos de código para la versión AWS SDK for Java 2.x](#)
- [Usar el SDK](#)
- [Seguridad para el AWS SDK for Java](#)

step-by-step Instrucciones de migración con ejemplo

En esta sección se proporciona una step-by-step guía para migrar la aplicación que actualmente usa el SDK para Java v1.x al SDK para Java 2.x. La primera parte presenta una descripción general de los pasos, seguida de un ejemplo detallado de una migración.

Los pasos que se describen aquí describen la migración de un caso de uso normal, en el que la aplicación llama Servicios de AWS mediante clientes de servicio basados en modelos. Si necesita migrar código que utilice API de nivel superior, como [S3 Transfer Manager](#) o la [CloudFrontpresignación](#), consulte la sección situada debajo del [the section called “Diferencias entre 1.x y 2.x”](#) índice.

El enfoque que se describe aquí es una sugerencia. Puede utilizar otras técnicas y aprovechar las funciones de edición de código de su IDE para obtener el mismo resultado.

Información general sobre los pasos

1. Comience por añadir la BOM de SDK for Java 2.x

Al añadir el elemento BOM (lista de materiales) de Maven para el SDK for Java 2.x a su archivo POM, se asegura de que todas las dependencias de la versión 2 que necesita son de la misma versión. Tu POM puede contener dependencias tanto en la versión 1 como en la versión 2. Esto le permite migrar el código de forma incremental en lugar de cambiarlo todo de una vez.

BOM de SDK for Java 2.x

```
<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>
```

Puede encontrar la [última versión en el repositorio](#) central de Maven.

2. Busque en los archivos las declaraciones de importación de la clase v1

Si escanea los archivos de su aplicación en busca de los Service_IDs utilizados en las importaciones de la versión 1, encontrará los Service_IDs únicos utilizados. Un SERVICE_ID es un nombre corto y único para un Servicio de AWS. Por ejemplo, `cognitoidentity` es el SERVICE_ID de Amazon Cognito Identity.

3. Determine las dependencias de Maven en la versión 2 a partir de las declaraciones de importación de la versión 1

Tras encontrar todos los Service_ID únicos de la versión 1, puede determinar el artefacto de Maven correspondiente a la dependencia de la versión 2 consultando [the section called “Nombre del paquete a las asignaciones de ArtifactsID”](#)

4. Añada los elementos de dependencia de la versión 2 al archivo POM

Actualice el archivo POM de Maven con los elementos de dependencia determinados en el paso 3.

5. En los archivos Java, cambie gradualmente las clases v1 a las clases v2

A medida que sustituya las clases de la versión 1 por las de la versión 2, realice los cambios necesarios para que sean compatibles con la API de la versión 2, como utilizar compiladores en lugar de constructores y utilizar captadores y configuradores fluidos.

6. Elimine las dependencias de Maven de la versión 1 del POM y las importaciones de la versión 1 de los archivos

Después de migrar el código para usar las clases de la versión 2, elimina las importaciones de la versión 1 sobrantes de los archivos y todas las dependencias del archivo de compilación.

7. Refactoriza el código para usar las mejoras de la API de la versión 2

Una vez que el código se haya compilado correctamente y superado las pruebas, puede aprovechar las mejoras de la versión 2, como utilizar un cliente HTTP diferente o paginadores para simplificar el código. Se trata de un paso opcional.

Ejemplo de migración

En este ejemplo, migramos una aplicación que usa el SDK for Java v1 y accede a varios Servicios de AWS. En el paso 5 analizaremos en detalle el siguiente método v1. Este es un método de una clase que contiene ocho métodos y hay 32 clases en la aplicación.

método v1 para migrar

A continuación, solo se enumeran las importaciones del SDK de la versión 1 del archivo 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.getInstanceIds()) {
                    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. Agregue la versión 2 de Maven BOM

Agregue la BOM de Maven para el SDK for Java 2.x al POM junto con cualquier otra dependencia de la sección. `dependencyManagement` Si tu archivo POM tiene la BOM de la versión 1 del SDK, déjala por ahora. Se eliminará en un paso posterior.

POM: Gestión de dependencias desde el principio

```

<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>
    ...
  </dependencies>
</dependencyManagement>

```

```
<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>
```

2. Busque en los archivos las declaraciones de importación de la clase v1

Busque en el código de la aplicación las apariciones únicas de `import com.amazonaws.services`. Esto nos ayuda a determinar las dependencias de la versión 1 que utiliza el proyecto. Si su aplicación tiene un archivo POM de Maven con las dependencias de la versión 1 en la lista, puede utilizar esta información en su lugar.

En este ejemplo, utilizamos el comando [ripgrep\(rg\)](#) para buscar en la base de código.

Desde la raíz de su base de código, ejecute el siguiente `ripgrep` comando. Una vez `ripgrep` que encuentra las sentencias de importación, se canalizan a los `uniq` comandos `cut` `sort`, y para aislar los `Service_IDS`.

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

Para esta aplicación, los siguientes `Service_IDS` se registran en la consola.

```
autoscaling
cloudformation
ec2
identitymanagement
```

Esto indica que se utilizó al menos una vez cada uno de los siguientes nombres de paquetes en `import` las sentencias. Para cumplir con nuestros propósitos, los nombres de las clases individuales no importan. Solo necesitamos encontrar los `Service_IDS` que se utilizan.

```
com.amazonaws.services.autoscaling.*
com.amazonaws.services.cloudformation.*
com.amazonaws.services.ec2.*
com.amazonaws.services.identitymanagement.*
```

3. Determine las dependencias de Maven en la versión 2 a partir de las declaraciones de importación de la versión 1

Los ID de servicio de la versión 1 que hemos aislado del paso 2 (por ejemplo, `autoscaling` y `cloudformation`) pueden asignar al mismo `SERVICE_ID` de la versión 2 en su mayor parte. Como el `ArtifactId` de Maven v2 coincide con el `SERVICE_ID` en la mayoría de los casos, tienes la información que necesitas para añadir bloques de dependencias a tu archivo POM.

La siguiente tabla muestra cómo podemos determinar las dependencias de la versión 2.

v1 SERVICE_ID se asigna a...	v2 SERVICE_ID se asigna a...	dependencia de Maven v2
nombre del paquete ec2 com.amazonaws.serv ices. ec2.*	nombre del paquete ec2 software.amazon.aw ssdk.services. ec2.*	<pre><dependency> <groupId>software. amazon.awssdk</gro upId> <artifactId> ec2</ artifactId> </dependency></pre>
escalado automático com.amazonaws.serv ices. autoscaling .*	escalado automático software.amazon.aw ssdk.serv ices. autoscaling .*	<pre><dependency> <groupId>software. amazon.awssdk</gro upId> <artifactId> autoscali ng </artifactId> </dependency></pre>
cloudformation com.amazonaws.serv ices. cloudform ation .*	cloudformation software.amazon.aw ssdk. cloudform ation .*	<pre><dependency> <groupId>software. amazon.awssdk</gro upId> <artifactId> cloudform ation </artifactId> </dependency></pre>
gestión de la identidad*	yo*	<pre><dependency></pre>

v1 SERVICE_ID se asigna a...	v2 SERVICE_ID se asigna a...	dependencia de Maven v2
nombre del paquete	nombre del paquete	
com.amazonaws.services. identitymanagement .*	software.amazon.awssdk. iam .*	<pre><groupId>software. amazon.awssdk</gro upId> <artifactId> iam</ artifactId> </dependency></pre>

* El iam mapeo identitymanagement to es una excepción en la que el SERVICE_ID difiere entre las versiones. Consulta las [the section called “Nombre del paquete a las asignaciones de ArtifactID”](#) excepciones si Maven o Gradle no pueden resolver la dependencia de la versión 2.

4. Agrega los elementos de dependencia de la versión 2 al archivo POM

En el paso 3, determinamos los cuatro bloques de dependencia que deben añadirse al archivo POM. No necesitamos añadir una versión porque especificamos la BOM en el paso 1. Una vez agregadas las importaciones, nuestro archivo POM tiene los siguientes elementos de dependencia.

```
...
<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. En los archivos Java, cambie gradualmente las clases v1 a las clases v2

En el método que estamos migrando, vemos

- Un cliente de servicio EC2 de `com.amazonaws.services.ec2.AmazonEC2Client`
- Se utilizaron varias clases de modelos EC2. Por ejemplo, `DescribeInstancesRequest` y `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.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;
}
...
```

Nuestro objetivo es reemplazar todas las importaciones de la versión 1 con las importaciones de la versión 2. Procedemos una clase a la vez.

a. Sustituya la declaración de importación o el nombre de la clase

Vemos que el primer parámetro del `describeRunningInstances` método es una `AmazonEC2Client` instancia v1. Realice una de las siguientes acciones siguientes:

- Sustituya la importación por `com.amazonaws.services.ec2.AmazonEC2Client` `software.amazon.awssdk.services.ec2.Ec2Client` y `AmazonEC2Client` cámbiela por `Ec2Client`.
- Cambie el tipo de parámetro a `Ec2Client` y deje que el IDE nos pida la importación correcta. Nuestro IDE nos pedirá que importe la clase v2 porque los nombres de los clientes son diferentes: `AmazonEC2Client` y `Ec2Client`. Este enfoque no funciona si el nombre de la clase es el mismo en ambas versiones.

b. Sustituya las clases del modelo v1 por equivalentes a la v2

Tras el cambio a la versión 2 `Ec2Client`, si utilizamos un IDE, vemos errores de compilación en la siguiente declaración.

```
result = ec2.describeInstances(request);
```

El error de compilación se debe al uso de una instancia de v1 `DescribeInstancesRequest` como parámetro del `Ec2Client` `describeInstances` método v2. Para solucionarlo, realiza las siguientes instrucciones de sustitución o importación.

replace	with
<pre>import com.amazonaws.services.ec2.model.DescribeInstancesRequest</pre>	<pre>import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest</pre>

c. Cambie los constructores v1 por constructores v2.

Seguimos viendo errores de compilación porque [no hay constructores en las clases de la versión 2](#). Para solucionarlo, realiza el siguiente cambio.

cambiar	a
<pre>final DescribeInstancesRequest request = new DescribeInstancesRequest().withInstanceIds(instanceIdsCopy);</pre>	<pre>final DescribeInstancesRequest request = DescribeInstancesRequest.builder().instanceIds(instanceIdsCopy).build();</pre>

d. Sustituya los objetos de ***Result** respuesta de la versión 1 por ***Response** equivalentes de la versión 2

Una diferencia constante entre v1 y v2 es que todos los [objetos de respuesta de v2 terminan en *Response lugar de *Result](#). Reemplace la `DescribeInstancesResult` importación de v1 por la importación de v2, `DescribeInstancesResponse`.

d. Realice cambios en la API

La siguiente declaración necesita algunos cambios.

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

En la versión 2, [los métodos setter](#) no utilizan `set` o `withprefix`. Los métodos Getter con `get` el prefijo de también han desaparecido en el SDK para Java 2.x

Las clases de modelos, como la `request` instancia, son inmutables en la versión 2, por lo que necesitamos crear una nueva con un generador. `DescribeInstancesRequest`

En la versión 2, la sentencia pasa a ser la siguiente.

```
request = DescribeInstancesRequest.builder()
    .nextToken(result.nextToken())
    .build();
```

d. Repita el procedimiento hasta que el método se compile con las clases de la versión 2

Continúe con el resto del código. Sustituya las importaciones de la versión 1 por las de la versión 2 y corrija los errores de compilación. Consulte la [referencia de la API](#) de la versión 2 y la [referencia Qué es diferente](#) según sea necesario.

Tras migrar este único método, tendremos el siguiente código de la versión 2.

```
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 un único método en un archivo Java con ocho métodos, tenemos una combinación de importaciones de la versión 1 y la versión 2 a medida que avanzamos en el archivo. Añadimos las seis últimas declaraciones de importación a medida que realizábamos los pasos.

Después de migrar todo el código, no habrá más declaraciones de importación de la versión 1.

6. Elimine las dependencias de Maven de la versión 1 del POM y las importaciones de la versión 1 de los archivos

Tras migrar todo el código de la versión 1 del archivo, tenemos las siguientes instrucciones de importación del SDK de la versión 2.

```
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;
```

Tras migrar todos los archivos de nuestra aplicación, ya no necesitamos las dependencias de la versión 1 en nuestro archivo POM. Elimine la lista de materiales de la versión 1 de la `dependencyManagement` sección, si la está utilizando, y todos los bloques de dependencias de la versión 1.

7. Refactoriza el código para utilizar las mejoras de la API de la versión 2

Para el fragmento que hemos estado migrando, podemos usar opcionalmente un paginador de la versión 2 y dejar que el SDK gestione las solicitudes de más datos basadas en fichas.

Podemos sustituir toda la cláusula por lo siguiente. `do`

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

```
    }
  }));
```

Nombre del paquete para Maven ArtifactID mappings

Al migrar tu proyecto de Maven o Gradle de la versión 1 del SDK para Java a la versión 2, debes determinar qué dependencias añadir al archivo de compilación. El enfoque descrito en el [the section called “tep-by-step Instrucciones S”](#) (paso 3) utiliza los nombres de los paquetes de las instrucciones de importación como punto de partida para determinar las dependencias (como ArtifactIDs) que se van a añadir al archivo de compilación.

Puedes usar la información de este tema para asignar los nombres de los paquetes de la versión 1 a los ArtifactIDs de la versión 2.

Convención de nomenclatura común utilizada en los nombres de paquetes y en los ArtifactID de Maven

En la siguiente tabla se muestra la convención de nomenclatura común que utilizan los SDK para un SERVICE_ID determinado. Un SERVICE_ID es un identificador único para un. Servicio de AWS Por ejemplo, el SERVICE_ID del servicio Amazon S3 es s3 y cognitoidentity es el SERVICE_ID de Amazon Cognito Identity.

Nombre del paquete v1 (declaración de importación)	1 ArtifactID	ArtifactID v2	nombre del paquete v2 (declaración de importación)
com.amazonaws.services.SERVICE_ID	aws-java-sdk-SERVICE_ID	SERVICE_ID	software.amazon.awssdk.services.service_id

Ejemplo de Amazon Cognito Identity (SERVICE_ID:) ***cognitoidentity***

com.amazonaws.services.identidad cognitiva	aws-java-sdk-identidad cognitiva	identidad cognitiva	software.amazon.awssdk.services.identidad cognitiva
--	----------------------------------	---------------------	---

Diferencias entre SERVICE_ID

Dentro de la versión 1

En algunos casos, el SERVICE_ID difiere entre el nombre del paquete y el ArtifactID del mismo servicio. Por ejemplo, la fila de CloudWatch métricas de la siguiente tabla muestra que `metrics` es el SERVICE_ID del nombre del paquete, pero es el SERVICE_ID del ArtifactID.

```
cloudwatchmetrics
```

Dentro de la versión 2

No hay diferencias en el SERVICE_ID utilizado en los nombres de los paquetes y en los ArtifactID.

Entre v1 y v2

En la mayoría de los servicios, el SERVICE_ID de la versión 2 es el mismo que el SERVICE_ID de la versión 1, tanto en los nombres de los paquetes como en los ArtifactID. Un ejemplo de esto es el `cognitoentity` SERVICE_ID, como se muestra en la tabla anterior. Sin embargo, algunos Service_ID difieren entre los SDK, como se muestra en la siguiente tabla.

Un SERVICE_ID en negrita en cualquiera de las columnas de la versión 1 indica que es diferente del SERVICE_ID utilizado en la versión 2.

Nombre del servicio	nombre del paquete v1	1 ArtifactID	ArtifactID v2	nombre del paquete v2
	Todos los nombres de los paquetes comienzan por <code>com.amazonservices</code> lo que se muestra en la primera fila.	Todos los ArtifactID están encerrados en etiquetas, como se muestra en la primera fila.	Todos los ArtifactID están encerrados en etiquetas, como se muestra en la primera fila.	Todos los nombres de los paquetes comienzan por <code>software.amazon.aws</code> lo que se muestra en la primera fila.
API Gateway	<code>com.amazonaws.apigateway</code>	<code><artifactId>aws-java-sdk-puerta</code>	<code><artifactId>una puerta de</code>	<code>puerta de enlace software.</code>

Nombre del servicio	nombre del paquete v1	1 ArtfactiD	ArtfactiD v2	nombre del paquete v2
		de enlace api</artifacId>	enlace</artifacId>	amazon.awssdk.services.api
Registro de aplicaciones	preregistro	apregistro/registro	catálogo de servicios y registro	registro de aplicaciones de catálogos de servicios
Application Discovery	descubrimiento de aplicaciones	discovery	descubrimiento de aplicaciones	descubrimiento de aplicaciones
Augmented AI Runtime	tiempo de ejecución de aire aumentado	aumento del tiempo de funcionamiento del aire	tiempo de ejecución de sagemaker a2	tiempo de ejecución de sagemakera2
Certificate Manager	administrador de certificados	acm	acm	acm
CloudControl API	api de control en la nube	api de control en la nube	control en la nube	control en la nube
CloudSearch	cloudsearch v2	cloudsearch	cloudsearch	cloudsearch
CloudSearch Dominio	dominio de búsqueda en la nube	búsqueda en la nube	dominio de búsqueda en la nube	dominio de búsqueda en la nube
CloudWatch Eventos	eventos de cloudwatch	eventos	eventos de cloudwatch	eventos de cloudwatch
CloudWatch Evidentemente	vigilancia en la nube, evidentemente	cloudwatch evidentemente	evidently	evidently

Nombre del servicio	nombre del paquete v1	1 ArtfactiD	ArtfactiD v2	nombre del paquete v2
CloudWatch Registros	logs	logs	registros de cloudwatch	registros de cloudwatch
CloudWatch Métricas	métricas	métricas de cloudwatch	cloudwatch	cloudwatch
CloudWatch Ron	cloudwatchrum	cloudwatchrum	rum	rum
Proveedor de identidad de Cognito	cognitoidp	cognitoide	proveedor de identidad cognito	proveedor de identidad cognito
Campaña Connect	conectar la campaña	conectar la campaña	conectar campañas	conectar campañas
Connect Wisdom	conecta la sabiduría	conecta la sabiduría	wisdom	wisdom
Database Migration Service	servicio de migración de bases de datos	dms	migración de bases de datos	migración de bases de datos
DataZone	zona de datos	zona de datos externa	zona de datos	zona de datos
DynamoDB	dynamodbv2	dynamodb	dynamodb	dynamodb
Sistema de archivos elástico	sistema de archivos elástico	efs	efs	efs
Elastic Map Reduce	elasticma preduce	emr	emr	emr
Glue DataBrew	pega una cerveza	gluedatabrew	databrew	databrew

Nombre del servicio	nombre del paquete v1	1 ArtfactiD	ArtfactiD v2	nombre del paquete v2
Funciones de IAM en cualquier lugar	soy roles en cualquier parte	soy roles en cualquier parte	rolesanywhere	rolesanywhere
Administración de identidades	gestión de identidad	iam	iam	iam
Datos de IoT	datos de IoT	iot	plano de datos de IoT	plano de datos de IoT
Análisis de Kinesis	kinesisanalytics	kinesis	kinesisanalytics	kinesisanalytics
Kinesis Firehose	manguera de incendios Kinesis	kinesis	firehose	firehose
Canales de señalización de vídeo de Kinesis	canales de señalización de vídeo de Kinesis	canales de señalización de vídeo de kinesis	señalización de vídeo de cinesia	señalización de vídeo de cinesia
Lex	tiempo de ejecución de lex	Alex	lexruntime	lexruntime
¡Busca la visión	busca la visión	busca la visión	lookoutvision	lookoutvision
Modernización del mainframe	modernización del mainframe	modernización del mainframe	m2	m2
Medición Marketplace	medición en el mercado	servicio de medición en el mercado	medición en el mercado	medición en el mercado
Grafana gestionada	grafana gestionado	grafana gestionada	grafana	grafana

Nombre del servicio	nombre del paquete v1	1 ArtifactsID	ArtifactsID v2	nombre del paquete v2
Mechanical Turk	mturk	solicitante mecánico de turcas	turk	mturk
Recomendaciones de estrategias de Migration Hub	recomendaciones de estrategia para el centro de migración	recomendaciones de estrategia para el centro de migración	estrategia de centro de migración	estrategia de centro de migración
Nimble Studio	estudio ágil	estudio ágil	nimble	nimble
5G privado	5G privado	5g privado	redes privadas	redes privadas
Prometheus	prometeo	prometeo	amperio	amperio
Papelera de reciclaje	papelera de reciclaje	papelera de reciclaje	rbin	rbin
API de datos de Redshift	redshift data api	redshift data api	datos de corrimiento al rojo	datos de corrimiento al rojo
Route 53	enrutar 53 dominios	route53	enrutar 53 dominios	enrutar 53 dominios
Administrador Edge de Sage Maker	administrador sagemaker edge	sagemakeredge manager	sagemakeredge	sagemakeredge
Token de seguridad	token de seguridad	sts	sts	sts
Migración de servidores	migración de servidores	migración de servidores	sms	sms

Nombre del servicio	nombre del paquete v1	1 ArtfactiD	ArtfactiD v2	nombre del paquete v2
Correo electrónico simple	correo electrónico simple	ses	ses	ses
Correo electrónico simple V2	correo electrónico simple v2	sesv2	sesv2	sesv2
Administración de sistemas sencilla	administración de sistemas sencilla	ssm	ssm	ssm
Flujo de trabajo sencillo	flujo de trabajo sencillo	flujo de trabajo simple	swf	swf
Step Functions	funciones escalonadas	funciones escalonadas	sfn	sfn

Qué diferencia hay entre la AWS SDK for Java 1.x y la 2.x

En esta sección se describen los principales cambios que hay que tener en cuenta al convertir una aplicación de la AWS SDK for Java versión 1.x a la versión 2.x.

Cambio de nombre de paquete

Un cambio notable del SDK para Java 1.x al SDK para Java 2.x es el cambio de nombre del paquete. Los nombres de los paquetes comienzan por `software.amazon.awssdk` en el SDK 2.x, mientras que en el SDK 1.x se usa `com.amazonaws`.

Estos mismos nombres diferencian los artefactos de Maven del SDK 1.x del SDK 2.x. Los artefactos de Maven para el SDK 2.x usan el GroupID `software.amazon.awssdk`, mientras que el SDK 1.x usa el GroupID `com.amazonaws`.

Hay ocasiones en las que el código requiere una dependencia `com.amazonaws` para un proyecto que, de otro modo, solo utilizaría artefactos del SDK 2.x. Un ejemplo de ello es cuando trabajas con AWS Lambda del lado del servidor. Esto se mostró anteriormente en la sección [Configurar un proyecto de Apache Maven](#) en esta guía.

Note

Varios nombres de paquetes en el SDK 1.x contienen v2. El uso de v2 en este caso suele significar que el código del paquete está orientado a funcionar con la versión 2 del servicio. Como el nombre completo del paquete comienza por `com.amazonaws`, se trata de componentes del SDK 1.x. Algunos ejemplos de estos nombres de paquetes en el SDK 1.x son:

- `com.amazonaws.services.dynamodbv2`
- `com.amazonaws.retry.v2`
- `com.amazonaws.services.apigatewayv2`
- `com.amazonaws.services.simpleemailv2`

Adición de la versión 2.x a su proyecto

Maven es la forma recomendada de gestionar las dependencias cuando se utiliza la 2.x. AWS SDK for Java Para añadir componentes de la versión 2.x a tu proyecto, actualiza tu `pom.xml` archivo con una dependencia del 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>
```

También puedes [usar las versiones 1.x y 2.x side-by-side](#) al migrar tu proyecto a la versión 2.x.

POJO inmutables

Los objetos de respuesta y solicitud de operación y clientes son ahora inmutables y no se pueden cambiar tras la creación. Para reutilizar una variable de respuesta o solicitud, debe crear un objeto nuevo para asignarlo a la misma.

Example de actualización de un objeto de solicitud en 1.x

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

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

Example de actualización de un objeto de solicitud en 2.x

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

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

Métodos Setter y Getter

En la versión AWS SDK for Java 2.x, los nombres de los métodos setter no incluyen el prefijo `or. set` `with`. Por ejemplo, `*.withEndpoint()` es ahora `*.endpoint()`.

Los nombres de los métodos Getter no utilizan el prefijo `get`.

Example de usar métodos setter en 1.x

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

Example de usar métodos setter en 2.x

```
DynamoDbClient client = DynamoDbClient.builder()
```

```
.region(Region.US_EAST_1)
.build();
```

Example de usar métodos getter en 1.x

```
String token = request.getNextToken();
```

Example de usar métodos de captación en la versión 2.x

```
String token = request.nextToken();
```

Nombres de clases de modelos

Los nombres de las clases de modelo que representan las respuestas de los servicios terminan `Response` en la versión 2 y no en los `Result` que utiliza la versión 1.

Example de nombres de clases que representan una respuesta en la v1

```
CreateApiKeyResult
AllocateAddressResult
```

Example de nombres de clases que representan una respuesta en la versión 2

```
CreateApiKeyResponse
AllocateAddressResponse
```

Estado migratorio de bibliotecas y utilidades

Bibliotecas y utilidades del SDK para Java

En la siguiente tabla, se muestra el estado de migración de las bibliotecas y utilidades del SDK para Java.

Nombre de la versión 1.12.x	Nombre de la versión 2.x	A partir de la versión 2.x
DynamoDBMapper	DynamoDbEnhancedClient	2.12.0
Esperadores	Esperadores	2.15.0

Nombre de la versión 1.12.x	Nombre de la versión 2.x	A partir de la versión 2.x
CloudFrontUrlSigner, CloudFrontCookieSigner	CloudFrontUtilities	2.18.33
TransferManager	S3 TransferManager	2.19.0
Cliente de metadatos EC2	Cliente de metadatos de EC2	2.19.29
Analizador de URI S3	Analizador de URI S3	2.20,41
Creador de políticas de IAM	Creador de políticas de IAM	2.20.126
Almacenamiento en búfer del cliente Amazon SQS	Procesamiento automático de solicitudes por lotes	no publicado aún
Agentes de escucha de progreso	Agentes de escucha de progreso	no publicado aún

Bibliotecas relacionadas

En la tabla siguiente se enumeran las bibliotecas que se publican por separado pero que funcionan con el SDK para Java 2.x.

Nombre utilizado en la versión 2.x del SDK para Java	Desde la versión
Cliente de cifrado de Amazon S3	3.0.0 1
AWS Cliente de cifrado de bases de datos para DynamoDB	3.0.0 2

¹El cliente de cifrado para Amazon S3 está disponible mediante la siguiente dependencia de Maven.

```
<dependency>
  <groupId>software.amazon.encryption.s3</groupId>
  <artifactId>amazon-s3-encryption-client-java</artifactId>
  <version>3.x</version>
```



```
</dependency>
```

² El cliente AWS de cifrado de bases de datos para DynamoDB está disponible mediante la siguiente dependencia de Maven.

```
<dependency>
  <groupId>software.amazon.cryptography</groupId>
  <artifactId>aws-database-encryption-sdk-dynamodb</artifactId>
  <version>3.x</version>
</dependency>
```

Detalles de migración para bibliotecas y utilidades

- [Administrador de transferencias de S3](#)
- [Utilidad de metadatos EC2](#)
- [CloudFront prefirmando](#)
- [análisis de URI de S3](#)

Cambios de cliente

Compiladores de clientes

Debe crear todos los clientes mediante el método del compilador de clientes. Los constructores ya no están disponibles.

Example de creación de un cliente en la versión 1.x

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

Example de creación de un cliente en la versión 2.x

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

Nombres de clases de clientes

Ahora todos los nombres de las clases de los clientes se escriben con mayúsculas y minúsculas y ya no llevan el prefijo Amazon. Estos cambios concuerdan con los nombres utilizados en la AWS CLI.

Exemple de nombres de clase en 1.x

```
AmazonDynamoDB
AWSACMPAAsyncClient
```

Exemple de nombres de clase en 2.x

```
DynamoDbClient
AcmAsyncClient
```

Cambios en el nombre de la clase de cliente

1.x Cliente	Cliente 2.x
<code>com.amazonaws.services.acmpca.AWSACMPAAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.acmpca.AWSACMPCAClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessAsyncClient</code>	<code>software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessAsyncClient</code>
<code>com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessClient</code>	<code>software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayAsyncClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayAsyncClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayClient</code>
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling</code>

1.x Cliente	Cliente 2.x
	<code>.ApplicationAutoScalingAsyncClient</code>
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient</code>
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryAsyncClient</code>
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryClient</code>
<code>com.amazonaws.services.appstream.AmazonAppStreamAsyncClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamAsyncClient</code>
<code>com.amazonaws.services.appstream.AmazonAppStreamClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamClient</code>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<code>com.amazonaws.services.cognitoidentityprovider.AWSCognitoIdentityProviderAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderAsyncClient</code>
<code>com.amazonaws.services.cognitoidentityprovider.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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<code>com.amazonaws.services.dax.AmazonDaxClient</code>	<code>software.amazon.awssdk.services.dax.DaxClient</code>
<code>com.amazonaws.services.devicefarm.AWSDeviceFarmAsyncClient</code>	<code>software.amazon.awssdk.services.devicefarm.DeviceFarmAsyncClient</code>
<code>com.amazonaws.services.devicefarm.AWSDeviceFarmClient</code>	<code>software.amazon.awssdk.services.devicefarm.DeviceFarmClient</code>
<code>com.amazonaws.services.directconnect.AmazonDirectConnectAsyncClient</code>	<code>software.amazon.awssdk.services.directconnect.DirectConnectAsyncClient</code>

1.x Cliente	Cliente 2.x
<code>com.amazonaws.services.directconnect.AmazonDirectConnectClient</code>	<code>software.amazon.awssdk.services.directconnect.DirectConnectClient</code>
<code>com.amazonaws.services.directory.AWSDirectoryServiceAsyncClient</code>	<code>software.amazon.awssdk.services.directory.DirectoryAsyncClient</code>
<code>com.amazonaws.services.directory.AWSDirectoryServiceClient</code>	<code>software.amazon.awssdk.services.directory.DirectoryClient</code>
<code>com.amazonaws.services.dlm.AmazonDLMAsyncClient</code>	<code>software.amazon.awssdk.services.dlm.DlmAsyncClient</code>
<code>com.amazonaws.services.dlm.AmazonDLMClient</code>	<code>software.amazon.awssdk.services.dlm.DlmClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient</code>
<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>

1.x Cliente	Cliente 2.x
<code>com.amazonaws.services.ecr. AmazonECRASyncClient</code>	<code>software.amazon.awssdk.serv ices.ecr.EcrAsyncClient</code>
<code>com.amazonaws.services.ecr. AmazonECRClient</code>	<code>software.amazon.awssdk.serv ices.ecr.EcrClient</code>
<code>com.amazonaws.services.ecs. AmazonECSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.ecs.EcsAsyncClient</code>
<code>com.amazonaws.services.ecs. AmazonECSClient</code>	<code>software.amazon.awssdk.serv ices.ecs.EcsClient</code>
<code>com.amazonaws.services.eks. AmazonEKSAyncClient</code>	<code>software.amazon.awssdk.serv ices.eks.EksAsyncClient</code>
<code>com.amazonaws.services.eks. AmazonEKSClient</code>	<code>software.amazon.awssdk.serv ices.eks.EksClient</code>
<code>com.amazonaws.services.elas ticache.AmazonElasticCacheAs yncClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElastiCach eAsyncClient</code>
<code>com.amazonaws.services.elas ticache.AmazonElasticCacheClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElastiCach eClient</code>
<code>com.amazonaws.services.elas ticbeanstalk.AWSElasticBean stalkAsyncClient</code>	<code>software.amazon.awssdk.serv ices.elasticbeanstalk.Elast icBeanstalkAsyncClient</code>
<code>com.amazonaws.services.elas ticbeanstalk.AWSElasticBean stalkClient</code>	<code>software.amazon.awssdk.serv ices.elasticbeanstalk.Elast icBeanstalkClient</code>
<code>com.amazonaws.services.elas ticfilesystem.AmazonElastic FileSystemAsyncClient</code>	<code>software.amazon.awssdk.serv ices.efs.EfsAsyncClient</code>

1.x Cliente	Cliente 2.x
<code>com.amazonaws.services.elasticfilesystem.AmazonElasticFileSystemClient</code>	<code>software.amazon.awssdk.services.efs.EfsClient</code>
<code>com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingAsyncClient</code>
<code>com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingClient</code>
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2AsyncClient</code>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>	No es compatible
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<code>com.amazonaws.services.mediataylor.AWSMediaTailorAsyncClient</code>	<code>software.amazon.awssdk.services.mediataylor.MediaTailorAsyncClient</code>
<code>com.amazonaws.services.mediataylor.AWSMediaTailorClient</code>	<code>software.amazon.awssdk.services.mediataylor.MediaTailorClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubAsyncClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubAsyncClient</code>
<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>

1.x Cliente	Cliente 2.x
<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>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMAsyncClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCMAsyncClient</code>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCMClient</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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<code>com.amazonaws.services.simp ledb.AmazonSimpleDBAsyncClient</code>	<code>software.amazon.awssdk.serv ices.simplesdb.SimpleDbAsync Client</code>
<code>com.amazonaws.services.simp ledb.AmazonSimpleDBClient</code>	<code>software.amazon.awssdk.serv ices.simplesdb.SimpleDbClient</code>
<code>com.amazonaws.services.simp leemail.AmazonSimpleEmailSe rviceAsyncClient</code>	<code>software.amazon.awssdk.serv ices.ses.SesAsyncClient</code>
<code>com.amazonaws.services.simp leemail.AmazonSimpleEmailSe rviceClient</code>	<code>software.amazon.awssdk.serv ices.ses.SesClient</code>
<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.SnowballAsync Client</code>
<code>com.amazonaws.services.snow ball.AmazonSnowballClient</code>	<code>software.amazon.awssdk.serv ices.snowball.SnowballClient</code>

1.x Cliente	Cliente 2.x
<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>
<code>com.amazonaws.services.step functions.AWSStepFunctionsA syncClient</code>	<code>software.amazon.awssdk.serv ices.sfn.SfnAsyncClient</code>
<code>com.amazonaws.services.step functions.AWSStepFunctionsC lient</code>	<code>software.amazon.awssdk.serv ices.sfn.SfnClient</code>
<code>com.amazonaws.services.stor agegateway.AWSStorageGatewa yAsyncClient</code>	<code>software.amazon.awssdk.serv ices.storagegateway.Storage GatewayAsyncClient</code>
<code>com.amazonaws.services.stor agegateway.AWSStorageGatewa yClient</code>	<code>software.amazon.awssdk.serv ices.storagegateway.Storage GatewayClient</code>
<code>com.amazonaws.services.supp ort.AWSSupportAsyncClient</code>	<code>software.amazon.awssdk.serv ices.support.SupportAsyncClient</code>
<code>com.amazonaws.services.supp ort.AWSSupportClient</code>	<code>software.amazon.awssdk.serv ices.support.SupportClient</code>
<code>com.amazonaws.services.tran scribe.AmazonTranscribeAsyn cClient</code>	<code>software.amazon.awssdk.serv ices.transcribe.TranscribeA syncClient</code>

1.x Cliente	Cliente 2.x
<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>
<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>

1.x Cliente	Cliente 2.x
<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>
<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>

Valores predeterminados de creación de clientes

En la versión 2.x, se han realizado los siguientes cambios en la lógica de creación de clientes predeterminada.

- La cadena de proveedores de credenciales predeterminada de S3 ya no incluye credenciales anónimas. Debe especificar manualmente el acceso anónimo a S3 mediante `AnonymousCredentialsProvider`
- Las siguientes variables de entorno relacionadas con la creación del cliente predeterminado son 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>

- Las siguientes propiedades del sistema relacionadas con la creación de clientes predeterminados son 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.disableIoBinary</code>	<code>aws.binaryIonEnabled</code>

- La versión 2.x no admite las siguientes propiedades del sistema.

-

1.x
<code>com.amazonaws.sdk.disableCertChecking</code>
<code>com.amazonaws.sdk.enableDefaultMetrics</code>
<code>com.amazonaws.sdk.enableThrottledRetry</code>
<code>com.amazonaws.regions.RegionUtils.fileOverride</code>
<code>com.amazonaws.regions.RegionUtils.disableRemote</code>
<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>
<code>com.amazonaws.sdk.enableInRegionOptimizedMode</code>

- Ya no se admite la carga de la configuración de la región desde un archivo `endpoints.json` personalizado.

Configuración de cliente

En la versión 1.x, la configuración del cliente del SDK se modificaba estableciendo una instancia `ClientConfiguration` en el cliente o en el compilador de clientes. En la versión 2.x, la configuración del cliente se divide en clases de configuración independientes. Con las clases de

configuración separadas, puede configurar diferentes clientes HTTP para clientes asíncronos frente a síncronos, pero seguir utilizando la misma clase `ClientOverrideConfiguration`.

Example de configuración del cliente en la versión 1.x

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

Example de configuración de clientes síncronos en la versión 2.x

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

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

ClientOverrideConfiguration.Builder overrideConfig =
    ClientOverrideConfiguration.builder();

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

Example de configuración de clientes asíncronos en la versión 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 de HTTP

Cambios notables

- En la versión 2.x, puede cambiar qué cliente HTTP usar en tiempo de ejecución especificando una implementación mediante `clientBuilder.httpClientBuilder`.
- Al pasar un cliente HTTP mediante `clientBuilder.httpClient` un generador de clientes de servicio, el cliente HTTP no se cierra de forma predeterminada si el cliente de servicio se cierra. Esto le permite compartir clientes HTTP entre clientes de servicio.
- Los clientes HTTP asíncronos ahora utilizan E/S sin bloqueo.
- Algunas operaciones ahora utilizan HTTP/2 para mejorar el rendimiento.

Cambios en la configuración

Opción	1.x	2.x Sync, Apache	2.x Async, 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 httpClient tBuilder = NettyNioA syncHttpC lient.builder()</pre>
Número máximo de conexiones	<pre>clientCon fig.setMa xConnecti ons(...) clientCon fig.withM axConnect ions(...)</pre>	<pre>httpClien tBuilder. maxConnec tions(...)</pre>	<pre>httpClien tBuilder. maxConcur rency(...)</pre>
Tiempo de espera de la conexión	<pre>clientCon fig.setCo</pre>	<pre>httpClien tBuilder.</pre>	<pre>httpClien tBuilder.</pre>

Opción	1.x	2.x Sync, Apache	2.x Async, Netty
	<pre> connectionTimeout(...) clientConfig.withConnectionTimeout(...) </pre>	<pre> connectionTimeout(...) httpClientBuilder.connectionAcquisitionTimeout(...) </pre>	<pre> connectionTimeout(...) </pre>
Tiempo de espera del enchufe	<pre> clientConfig.setSocketTimeout(...) clientConfig.withSocketTimeout(...) </pre>	<pre> httpClientBuilder.socketTimeout(...) </pre>	<pre> httpClientBuilder.writeTimeout(...) httpClientBuilder.readTimeout(...) </pre>
Conexión TTL	<pre> clientConfig.setConnectionTTL(...) clientConfig.withConnectionTTL(...) </pre>	<pre> httpClientBuilder.connectionTimeToLive(...) </pre>	<pre> httpClientBuilder.connectionTimeToLive(...) </pre>
Conexión máxima inactiva	<pre> clientConfig.setConnectionMaxIdleMillis(...) clientConfig.withConnectionMaxIdleMillis(...) </pre>	<pre> httpClientBuilder.connectionMaxIdleTime(...) </pre>	<pre> httpClientBuilder.connectionMaxIdleTime(...) </pre>

Opción	1.x	2.x Sync, Apache	2.x Async, Netty
Validar después de la inactividad	<pre>clientConfig.setValidateAfterInactivityMillis(...) clientConfig.withValidateAfterInactivityMillis(...)</pre>	No se admite (función de solicitud)	No se admite (función de solicitud)
Dirección local	<pre>clientConfig.setLocalAddress(...) clientConfig.withLocalAddress(...)</pre>	<pre>httpClientBuilder.localAddress(...)</pre>	No compatible
Activado el modo Expect-continue	<pre>clientConfig.setUseExpectContinue(...) clientConfig.withUseExpectContinue(...)</pre>	<pre>httpClientBuilder.expectContinueEnabled(...)</pre>	No se admite (función de solicitud)
Connection Reaper	<pre>clientConfig.setUseReaper(...) clientConfig.withReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>

Opción	1.x	2.x Sync, Apache	2.x Async, Netty
	<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

Configuración	1.x	2.x Sync, Apache	2.x Async, Netty
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>
Host del proxy.	<pre>clientConfig.setProxyHost(...) clientConfig.withProxyHost(...)</pre>	<pre>proxyConfig.endpoint(...)</pre>	<pre>proxyConfig.host(...)</pre>
Puerto del proxy.	<pre>clientConfig.setProxyPort(...) clientConfig.withProxyPort(...)</pre>	<pre>proxyConfig.endpoint(...)</pre>	<pre>proxyConfig.port(...)</pre>

Configuración	1.x	2.x Sync, Apache	2.x Async, Netty
		El puerto proxy está integrado endpoint	
Nombre de usuario del proxy	<pre>clientConfig.setProxyUsername(...) clientConfig.withProxyUsername(...)</pre>	<pre>proxyConfig.username(...)</pre>	<pre>proxyConfig.username(...)</pre>
Contraseña del proxy.	<pre>clientConfig.setProxyPassword(...) clientConfig.withProxyPassword(...)</pre>	<pre>proxyConfig.password(...)</pre>	<pre>proxyConfig.password(...)</pre>
Dominio proxy	<pre>clientConfig.setProxyDomain(...) clientConfig.withProxyDomain(...)</pre>	<pre>proxyConfig.ntlmDomain(...)</pre>	No compatible (función de solicitud)
Estación de trabajo proxy	<pre>clientConfig.setProxyWorkspace(...) clientConfig.withProxyWorkstation(...)</pre>	<pre>proxyConfig.ntlmWorkstation(...)</pre>	No compatible (función de solicitud)

Configuración	1.x	2.x Sync, Apache	2.x Async, Netty
Métodos de autenticación mediante proxy	<pre>clientConfig.setProxyAuthenticationMethods(...) clientConfig.withProxyAuthenticationMethods(...)</pre>	<u>No se admite</u>	No compatible (<u>función de solicitud</u>)
Autenticación proxy básica preventiva	<pre>clientConfig.setPreemptiveBasicProxyAuth(...) clientConfig.withPreemptiveBasicProxyAuth(...)</pre>	<pre>proxyConfig.preemptiveBasicAuthenticationEnabled(...)</pre>	No se admite (función de <u>solicitud</u>)
Hosts que no son proxy	<pre>clientConfig.setNonProxyHosts(...) clientConfig.withNonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>

Configuración	1.x	2.x Sync, Apache	2.x Async, Netty
Deshabilite el proxy de socket	<pre>clientConfig.setDisableSocketProxy(...) clientConfig.withDisableSocketProxy(...)</pre>	No compatible (función de solicitud)	No se admite (función de solicitud)
	<pre>AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>

El cliente anula

Opción	1.x	2.x
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>ClientOverrideConfiguration.Builder overrideConfig = ClientOverrideConfiguration.builder()</pre>
Prefijo del agente de usuario	<pre>clientConfig.setUserAgentPrefix(...) clientConfig.withUserAgentPrefix(...)</pre>	<pre>overrideConfig.advancedOption(SdkAdvancedClientOption.USER_AGENT_PREFIX, ...)</pre>

Opción	1.x	2.x
Sufijo de agente de usuario	<pre>clientConfig.setUserAgentSuffix(...) clientConfig.withUserAgentSuffix(...)</pre>	<pre>overrideConfig.advancedOption(SdkAdvancedClientOption.USER_AGENT_SUFFIX, ...)</pre>
Signer	<pre>clientConfig.setSignerOverride(...) clientConfig.withSignerOverride(...)</pre>	<pre>overrideConfig.advancedOption(SdkAdvancedClientOption.SIGNER, ...)</pre>
Encabezados adicionales	<pre>clientConfig.addHeader(...) clientConfig.withHeader(...)</pre>	<pre>overrideConfig.putHeader(...)</pre>
Tiempo de espera de la solicitud	<pre>clientConfig.setRequestTimeout(...) clientConfig.withRequestTimeout(...)</pre>	<pre>overrideConfig.apiCallAttemptTimeout(...)</pre>
Tiempo de espera de ejecución del cliente	<pre>clientConfig.setClientExecutionTimeout(...) clientConfig.withClientExecutionTimeout(...)</pre>	<pre>overrideConfig.apiCallTimeout(...)</pre>
Usa Gzip	<pre>clientConfig.setUseGzip(...) clientConfig.withGzip(...)</pre>	No compatible (función de solicitud)

Opción	1.x	2.x
Sugerencia de tamaño de búfer de socket	<pre>clientConfig.setSocketBufferSizeHints(...) clientConfig.withSocketBufferSizeHints(...)</pre>	No compatible (función de solicitud)
Guarda en caché los metadatos de respuesta	<pre>clientConfig.setCacheResponseMetadata(...) clientConfig.withCacheResponseMetadata(...)</pre>	No se admite (función de solicitud)
Tamaño de la caché de metadatos de respuesta	<pre>clientConfig.setResponseMetadataCacheSize(...) clientConfig.withResponseMetadataCacheSize(...)</pre>	No se admite (función de solicitud)
servicio de resolución de nombres DNS	<pre>clientConfig.setDnsResolver(...) clientConfig.withDnsResolver(...)</pre>	No se admite (función de solicitud)
TCP Keep Alive	<pre>clientConfig.setUseTcpKeepAlive(...) clientConfig.withTcpKeepAlive(...)</pre>	<p>Esta opción se encuentra ahora en la configuración del cliente HTTP</p> <pre>- ApacheHttpClient.builder().tcpKeepAlive(true) - NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)</pre>

Opción	1.x	2.x
Segura aleatoria	<pre>clientConfig.setSecureRandom(...) clientConfig.withSecureRandom(...)</pre>	No compatible (función de solicitud)
	<pre>AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>DynamoDbClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>

El cliente anula los reintentos

Opción	1.x	2.x
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>RetryPolicy.Builder retryPolicy = RetryPolicy.builder()</pre>
Error máximo: reintento	<pre>clientConfig.setMaxErrorRetry(...) clientConfig.withMaxErrorRetry(...)</pre>	<pre>retryPolicy.numRetries(...)</pre>
Utilice reintentos limitados	<pre>clientConfig.setUseThrottleRetries(...) clientConfig.withUseThrottleRetries(...)</pre>	No compatible
Máximo de reintentos consecutivos antes de limitar	<pre>clientConfig.setMaxConsecutiveRetrie</pre>	No compatible

Opción	1.x	2.x
	<pre>sBeforeThrottling(...) clientConfig.withMaxCo nsecutiveRetriesBe foreThrottling(...)</pre>	
	<pre>AmazonDynamoDBClie ntBuilder.standard() .withClientConfigu ration(clientConfi guration) .build()</pre>	<pre>DynamoDbClient.bui lder() .httpClientBuilder (httpClientBuilder) .build()</pre>

Cientes asíncronos

Opción	1.x	2.x
		<pre>ClientAsyncConfigu ration.Builder asyncConfig = ClientAsyncConfigu ration.builder()</pre>
Ejecutor	<pre>AmazonDynamoDBAsyn cClientBuilder.sta ndard() .withExecutorFacto ry(...) .build()</pre>	<pre>asyncConfig.advanc edOption(SdkAdvancedAsynCli entOption.FUTURE_ COMPLETION_EXECUTO R, ...)</pre>
		<pre>DynamoDbAsynClien t.builder() .asyncConfiguratio n(asyncConfig) .build()</pre>

Otros cambios en el cliente

La siguiente `ClientConfiguration` opción de la versión 1.x ha cambiado en la versión 2.x del SDK y no tiene un equivalente directo.

Opción	1.x	Equivalente a 2.x
Protocolo	<pre>clientConfig.setProtocol(Protocol.HTTP) clientConfig.withProtocol(Protocol.HTTP)</pre>	<p>La configuración del protocolo es HTTPS de forma predeterminada. Para modificar la configuración, especifique la configuración del protocolo como punto final HTTP en el generador de clientes:</p> <pre>clientBuilder.endpointOverride(URI.create("http://..."))</pre>

Cambios en el proveedor de credenciales

En esta sección se ofrece un esquema de los cambios de nombre de las clases y los métodos de proveedor de credenciales entre las versiones 1.x y 2.x del AWS SDK for Java.

Diferencias notables

- En la versión 2.x, el proveedor de credenciales predeterminado carga las propiedades del sistema antes que las variables de entorno. Para obtener más información, consulte [Uso de credenciales](#).
- El método constructor se sustituye por los métodos `create` o `builder`.

Example

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

- La actualización asíncrona ya no está configurada de forma predeterminada. Debe especificarla con el `builder` del proveedor de credenciales.

Example

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

- Puede especificar una ruta a un archivo de perfil personalizado utilizando `ProfileCredentialsProvider.builder()`.

Example

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

- El formato del archivo de perfil ha cambiado para que coincida mejor con la AWS CLI. Para obtener más información, consulte [Configurar la AWS CLI](#) en la Guía del usuario de AWS Command Line Interface.

Cambios del proveedor de credenciales entre las versiones 1.x y 2.x

AWSCredentialsProvider

Cambiar categoría	1.x	2.x
Nombre del paquete/clase	<code>com.amazonaws.auth.AWSCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.AwsCredentialsProvider</code>
Nombre del método	<code>getCredentials</code>	<code>resolveCredentials</code>
Método no compatible	<code>refresh</code>	No compatible

DefaultAWSCredentialsProviderChain

Cambiar categoría	1.x	2.x
Nombre del paquete/clase	<code>com.amazonaws.auth.DefaultAWSCredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code>
Creación	<code>new DefaultAWSCredentialsProviderChain</code>	<code>DefaultCredentialsProvider.create</code>
Método no compatible	<code>getInstance</code>	No compatible
Orden de prioridad de los ajustes externos	Las variables de entorno antes que las propiedades del sistema	Las propiedades del sistema antes que las variables de entorno

AWSStaticCredentialsProvider

Cambiar categoría	1.x	2.x
Nombre del paquete/clase	<code>com.amazonaws.auth.AWSStaticCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.StaticCredentialsProvider</code>
Creación	<code>new AWSStaticCredentialsProvider</code>	<code>StaticCredentialsProvider.create</code>

EnvironmentVariableCredentialsProvider

Cambiar categoría	1.x	2.x
Nombre del paquete/clase	<code>com.amazonaws.auth.EnvironmentVariable</code>	<code>software.amazon.awssdk.auth.credenti</code>

Cambiar categoría	1.x	2.x
	<code>EnvironmentVariableCredentialsProvider</code>	<code>aws.EnvironmentVariableCredentialsProvider</code>
Creación	<code>new EnvironmentVariableCredentialsProvider</code>	<code>EnvironmentVariableCredentialsProvider.create</code>
Nombre de la variable de entorno	<code>AWS_ACCESS_KEY</code>	<code>AWS_ACCESS_KEY_ID</code>
	<code>AWS_SECRET_KEY</code>	<code>AWS_SECRET_ACCESS_KEY</code>

SystemPropertiesCredentialsProvider

Cambiar categoría	1.x	2.x
Nombre del paquete/clase	<code>com.amazonaws.auth.SystemPropertiesCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.SystemPropertyCredentialsProvider</code>
Creación	<code>new SystemPropertiesCredentialsProvider</code>	<code>SystemPropertiesCredentialsProvider.create</code>
Nombre de propiedad del sistema	<code>aws.secretKey</code>	<code>aws.secretAccessKey</code>

ProfileCredentialsProvider

Cambiar categoría	1.x	2.x
Nombre del paquete/clase	<code>com.amazonaws.auth.profile.ProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider</code>
Creación	<code>new ProfileCredentialsProvider</code>	<code>ProfileCredentialsProvider.create</code>
Ubicación del perfil personalizado	<ul style="list-style-type: none"> • <code>AWS_CREDENTIAL_PROFILE_FILES_FILE</code> variable de entorno • <code>new ProfileCredentialsProvider</code> 	<ul style="list-style-type: none"> • <code>AWS_SHARED_CREDENTIALS_FILE</code> variable de entorno • <code>ProfileCredentialsProvider.builder</code>

ContainerCredentialsProvider

Cambiar categoría	1.x	2.x
Nombre del paquete/clase	<code>com.amazonaws.auth.ContainerCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code>
Creación	<code>new ContainerCredentialsProvider</code>	<code>ContainerCredentialsProvider.create</code>
Especifique la actualización asíncrona	No compatible	Comportamiento predeterminado

InstanceProfileCredentialsProvider

Cambiar categoría	1.x	2.x
Nombre del paquete/clase	<code>com.amazonaws.auth.InstanceProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
Creación	<code>new InstanceProfileCredentialsProvider</code>	<code>InstanceProfileCredentialsProvider.create</code>
Especifique la actualización asíncrona	<code>new InstanceProfileCredentialsProvider(true)</code>	<code>InstanceProfileCredentialProvider.builder().asyncCredentialUpdateEnabled(true).build()</code>
Nombre de propiedad del sistema	<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>

STSAssumeRoleSessionCredentialsProvider

Cambiar categoría	1.x	2.x
Nombre del paquete/clase	<code>com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider</code>

Cambiar categoría	1.x	2.x
Creación	<ul style="list-style-type: none"> <code>new STSAssumeRoleSessionCredentialsProvider</code> <code>new STSAssumeRoleSessionCredentialsProvider.Builder</code> 	<code>StsAssumeRoleCredentialsProvider.builder</code>
Actualización asíncrona	Comportamiento predeterminado	Comportamiento predeterminado
Configuración	<code>new STSAssumeRoleSessionCredentialsProvider.Builder</code>	Configure una solicitud de mano <code>StsClientAssumeRoleRequest</code>

STSSessionCredentialsProvider

Cambiar categoría	1.x	2.x
Nombre del paquete/clase	<code>com.amazonaws.auth.STSSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider</code>
Creación	<code>new STSAssumeRoleSessionCredentialsProvider</code>	<code>StsGetSessionTokenCredentialsProvider.builder</code>
Actualización asíncrona	Comportamiento predeterminado	<code>StsGetSessionTokenCredentialsProvider.builder</code>

Cambiar categoría	1.x	2.x
Configuración	Parámetros del constructor	Configure una <code>GetSessionTokenRequest</code> solicitud <code>StsClient</code> y en un generador

WebIdentityFederationSessionCredentialsProvider

Cambiar categoría	1.x	2.x
Nombre del paquete/clase	<code>com.amazonaws.auth.WebIdentityFederationSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProvider</code>
Creación	<code>new WebIdentityFederationSessionCredentialsProvider</code>	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>
Actualización asíncrona	Comportamiento predeterminado	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>
Configuración	Parámetros del constructor	Configure una <code>AssumeRoleWithWebIdentityRequest</code> solicitud <code>StsClient</code> y en un generador

Clases reemplazadas

Clase 1.x	2.x clases de reemplazo
<code>com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code> y <code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
<code>com.amazonaws.services.s3.S3CredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code> y <code>software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider</code>

Clases eliminadas

Clase 1.x
<code>com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider</code>
<code>com.amazonaws.auth.PropertiesFileCredentialsProvider</code>

Cambios de región

En esta sección se describen los cambios implementados en el AWS SDK for Java 2.x para usar las clases `Region` y `Regions`.

Configuración de región

- Algunos servicios de AWS no tienen puntos de conexión específicos de cada región. Al utilizar dichos servicios, debe configurar la región como `Region.AWS_GLOBAL` o `Region.AWS_CN_GLOBAL`.

Example

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

- Las clases `com.amazonaws.regions.Regions` y `com.amazonaws.regions.Region` se combinan ahora en una clase, `software.amazon.awssdk.regions.Region`.

Asignaciones de nombres de métodos y clases

Las tablas siguientes asignan clases relacionadas con la región entre las versiones 1.x y 2.x del AWS SDK for Java. Puede crear una instancia de estas clases utilizando el método `of()`.

Example

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

1.x Cambios en el método de las clases de Regions

1.x	2.x
<code>Regions.fromName</code>	<code>Region.of</code>
<code>Regions.getName</code>	<code>Region.id</code>
<code>Regions.getDescription</code>	<code>Region.metadata().description()</code>
<code>Regions.getCurrentRegion</code>	No es compatible
<code>Regions.DEFAULT_REGION</code>	No es compatible
<code>Regions.name</code>	<code>Region.id</code>

1.x Cambios en el método de las clases de regiones

1.x	2.x
<code>Region.getName</code>	<code>Region.id</code>
<code>Region.hasHttpsEndpoint</code>	No es compatible

1.x	2.x
<code>Region.hasHttpEndpoint</code>	No es compatible
<code>Region.getAvailableEndpoints</code>	No es compatible
<code>Region.createClient</code>	No es compatible

RegionMetadata cambios en el método de clase

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 cambios en el método de clase

1.x	2.x
<code>Region.getServiceEndpoint</code>	<code>ServiceMetadata.endpointFor(Region)</code>
<code>Region.isServiceSupported</code>	<code>ServiceMetadata.regions().contains(Region)</code>

Cambios en las operaciones, las solicitudes y las respuestas

En la versión 2.x del SDK for Java, las solicitudes se pasan a una operación de cliente. Por ejemplo, `DynamoDbClient's PutItemRequest` se pasa a `DynamoDbClient.putItem` la operación. Estas operaciones devuelven una respuesta de Servicio de AWS, como un `PutItemResponse`.

La versión 2.x del SDK para Java presenta los siguientes cambios con respecto a la versión 1.x.

- Las operaciones con varias páginas de respuesta ahora tienen un `Paginator` método para iterar automáticamente todos los elementos de la respuesta.

- No se pueden mutar las solicitudes y las respuestas.
- Debe crear solicitudes y respuestas con un método generador estático en lugar de un constructor. Por ejemplo, el de 1.x `new PutItemRequest().withTableName(...)` es ahora `PutItemRequest.builder().tableName(...).build()`.
- Las operaciones admiten una forma abreviada de crear solicitudes:
`dynamoDbClient.putItem(request -> request.tableName(...))`

Operaciones de streaming

Las operaciones de streaming, como Amazon S3 `getObject` y `putObject` los métodos, ahora admiten E/S sin bloqueo. Como resultado, los POJO de solicitud y respuesta ya no toman un `InputStream` como parámetro. En cambio, en el caso de las solicitudes sincrónicas, el objeto de solicitud acepta `RequestBody`, que es un flujo de bytes. El equivalente asíncrono acepta un `AsyncRequestBody`

Example de la operación **putObject** de Amazon S3 en 1.x

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

Example de la operación **putObject** de Amazon S3 en 2.x

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

En paralelo, un objeto de respuesta de transmisión acepta a `ResponseTransformer` para clientes sincrónicos y a `AsyncResponseTransformer` para clientes asíncronos.

Example de la operación **getObject** de Amazon S3 en 1.x

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

Example de la operación **getObject** de Amazon S3 en 2.x

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),
```



```
ResponseTransformer.toFile(Paths.get("key")));
```

En el SDK para Java 2.x, las operaciones de respuesta de streaming tienen un `AsBytes` método para cargar la respuesta en la memoria y simplificar las conversiones de tipos habituales en memoria.

Cambios de excepción

Los nombres de las clases de excepciones, sus estructuras y sus relaciones han cambiado. `software.amazon.awssdk.core.exception.SdkException` es la nueva `Exception` clase base a la que se extienden todas las demás excepciones.

En esta tabla se mapean los cambios de nombres de clase de excepción.

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>

En la siguiente tabla se mapean los métodos de las clases de excepción entre la versión 1.x y la 2.x.

1.x	2.x
<code>AmazonServiceException.getRequestId</code>	<code>SdkServiceException.requestId</code>
<code>AmazonServiceException.getServiceName</code>	<code>AwsServiceException.awsErrorDetails().serviceName</code>

1.x	2.x
<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>
<code>AmazonServiceException.getRawResponse</code>	<code>AwsServiceException.awsErrorDetails().rawResponse</code>

Cambios de serialización

Los SDK para Java v1.x y v2.x difieren en la forma en que serializan los objetos de lista para solicitar parámetros.

El SDK para Java 1.x no serializa una lista vacía, mientras que el SDK para Java 2.x serializa una lista vacía como un parámetro vacío.

Por ejemplo, pensemos en un servicio con un `SampleOperation` que toma un `SampleRequest`. `SampleRequest` acepta dos parámetros: un tipo de cadena `str1` y un tipo de lista `listParam`, como se muestra en los siguientes ejemplos.

Example de **SampleOperation** en 1.x

```
SampleRequest v1Request = new SampleRequest()
    .withStr1("TestName");

sampleServiceV1Client.sampleOperation(v1Request);
```

El registro a nivel de cable muestra que el parámetro `listParam` no está serializado.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName
```

Example de **SampleOperation** en 2.x

```
sampleServiceV2Client.sampleOperation(b -> b
    .str1("TestName"));
```

El registro a nivel de cable muestra que el parámetro `listParam` está serializado con ningún valor.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName&listParam=
```

Cambios específicos de los servicios

Cambios en Amazon S3

El SDK for Java 2.x deshabilita el acceso anónimo de forma predeterminada. Por lo tanto, debe habilitar el acceso anónimo mediante `AnonymousCredentialsProvider`

El nombre de la operación cambia

Muchos de los nombres de operación para el cliente de Amazon S3 han cambiado en el AWS SDK for Java 2.x. En la versión 1.x, el cliente de Amazon S3 no se generaba directamente desde la API de servicio. Esto se traduce en una incoherencia entre las operaciones de SDK y la API de servicio. En la versión 2.x, ahora generamos el cliente de Amazon S3 para que sea más coherente con la API de servicio.

En la siguiente tabla se muestran los nombres de las operaciones en las dos versiones.

Nombres de operaciones de 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>

1.x	2.x
<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>
<code>deleteBucketInventoryConfiguration</code>	<code>deleteBucketInventoryConfiguration</code>
<code>deleteBucketLifecycleConfiguration</code>	<code>deleteBucketLifecycle</code>
<code>deleteBucketMetricsConfiguration</code>	<code>deleteBucketMetricsConfiguration</code>
<code>deleteBucketPolicy</code>	<code>deleteBucketPolicy</code>
<code>deleteBucketReplicationConfiguration</code>	<code>deleteBucketReplication</code>
<code>deleteBucketTaggingConfiguration</code>	<code>deleteBucketTagging</code>
<code>deleteBucketWebsiteConfiguration</code>	<code>deleteBucketWebsite</code>
<code>deleteObject</code>	<code>deleteObject</code>
<code>deleteObjectTagging</code>	<code>deleteObjectTagging</code>
<code>deleteObjects</code>	<code>deleteObjects</code>
<code>deleteVersion</code>	<code>deleteObject</code>
<code>disableRequesterPays</code>	<code>putBucketRequestPayment</code>

1.x	2.x
doesBucketExist	headBucket
doesBucketExistV2	headBucket
doesObjectExist	headObject
enableRequesterPays	putBucketRequestPayment
generatePresignedUrl	S3Presigner
getBucketAccelerateConfiguration	getBucketAccelerateConfiguration
getBucketAcl	getBucketAcl
getBucketAnalyticsConfiguration	getBucketAnalyticsConfiguration
getBucketCrossOriginConfiguration	getBucketCors
getBucketEncryption	getBucketEncryption
getBucketInventoryConfiguration	getBucketInventoryConfiguration
getBucketLifecycleConfiguration	getBucketLifecycle o getBucketLifecycleConfiguration
getBucketLocation	getBucketLocation
getBucketLoggingConfiguration	getBucketLogging
getBucketMetricsConfiguration	getBucketMetricsConfiguration
getBucketNotificationConfiguration	getBucketNotification o getBucketNotificationConfiguration
getBucketPolicy	getBucketPolicy
getBucketReplicationConfiguration	getBucketReplication

1.x	2.x
<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>
<code>getObjectMetadata</code>	<code>headObject</code>
<code>getObjectTagging</code>	<code>getObjectTagging</code>
<code>getResourceUrl</code>	S3Utilities#getUrl
<code>getS3AccountOwner</code>	<code>listBuckets</code>
<code>getUrl</code>	S3Utilities#getUrl
<code>headBucket</code>	<code>headBucket</code>
<code>initiateMultipartUpload</code>	<code>createMultipartUpload</code>
<code>isRequesterPaysEnabled</code>	<code>getBucketRequestPayment</code>
<code>listBucketAnalyticsConfigurations</code>	<code>listBucketAnalyticsConfigurations</code>
<code>listBucketInventoryConfigurations</code>	<code>listBucketInventoryConfigurations</code>
<code>listBucketMetricsConfigurations</code>	<code>listBucketMetricsConfigurations</code>
<code>listBuckets</code>	<code>listBuckets</code>
<code>listMultipartUploads</code>	<code>listMultipartUploads</code>

1.x	2.x
<code>listNextBatchOfObjects</code>	<code>listObjectsV2Paginator</code>
<code>listNextBatchOfVersions</code>	<code>listObjectVersionsPaginator</code>
<code>listObjects</code>	<code>listObjects</code>
<code>listObjectsV2</code>	<code>listObjectsV2</code>
<code>listParts</code>	<code>listParts</code>
<code>listVersions</code>	<code>listObjectVersions</code>
<code>putObject</code>	<code>putObject</code>
<code>restoreObject</code>	<code>restoreObject</code>
<code>restoreObjectV2</code>	<code>restoreObject</code>
<code>selectObjectContent</code>	<code>selectObjectContent</code>
<code>setBucketAccelerateConfiguration</code>	<code>putBucketAccelerateConfiguration</code>
<code>setBucketAcl</code>	<code>putBucketAcl</code>
<code>setBucketAnalyticsConfiguration</code>	<code>putBucketAnalyticsConfiguration</code>
<code>setBucketCrossOriginConfiguration</code>	<code>putBucketCors</code>
<code>setBucketEncryption</code>	<code>putBucketEncryption</code>
<code>setBucketInventoryConfiguration</code>	<code>putBucketInventoryConfiguration</code>
<code>setBucketLifecycleConfiguration</code>	<code>putBucketLifecycle</code> o <code>putBucketLifecycleConfiguration</code>
<code>setBucketLoggingConfiguration</code>	<code>putBucketLogging</code>
<code>setBucketMetricsConfiguration</code>	<code>putBucketMetricsConfiguration</code>

1.x	2.x
setBucketNotificationConfiguration	putBucketNotification o putBucketNotificationConfiguration
setBucketPolicy	putBucketPolicy
setBucketReplicationConfiguration	putBucketReplication
setBucketTaggingConfiguration	putBucketTagging
setBucketVersioningConfiguration	putBucketVersioning
setBucketWebsiteConfiguration	putBucketWebsite
setObjectAcl	putObjectAcl
setObjectRedirectLocation	copyObject
setObjectTagging	putObjectTagging
uploadPart	uploadPart

Cambios en Amazon SNS

Un cliente de SNS ya no puede acceder a los temas de SNS en regiones distintas de la región para la que está configurado.

Cambios en Amazon SQS

Un cliente de SQS ya no puede acceder a las colas de SQS en regiones distintas de la región para la que está configurado.

Cambios en Amazon RDS

El SDK para Java 2.x se utiliza `RdsUtilities#generateAuthenticationToken` en lugar de la clase `RdsIamAuthTokenGenerator` de la versión 1.x.

Cambios en el archivo de perfil

AWS SDK for Java 2.x Analiza las definiciones de perfil en `~/.aws/config` y `~/.aws/credentials` para emular más de cerca la forma en que la AWS CLI analiza los archivos.

El SDK para Java 2.x:

- Resuelve un `~/` o `~` seguido del separador de rutas predeterminado del sistema de archivos al principio de la ruta comprobando, en orden, `$USERPROFILE` (solo en Windows), `$HOME`, `$HOMEDRIVE`, `$HOMEPATH` (solo en Windows) y, a continuación, la propiedad del sistema `user.home`.
- Busca la variable de entorno `AWS_SHARED_CREDENTIALS_FILE` en lugar de `AWS_CREDENTIAL_PROFILES_FILE`.
- Elimina silenciosamente las definiciones de perfil en los archivos de configuración sin incluir la palabra `profile` al principio del nombre del perfil.
- Elimina silenciosamente las definiciones de perfil que no estén compuestas por caracteres alfanuméricos, guiones bajos o guiones (una vez eliminada la `profile` palabra principal de los archivos de configuración).
- Combina los ajustes de las definiciones de perfil duplicadas en el mismo archivo.
- Combina los ajustes de las definiciones de perfil duplicados en los archivos de configuración y de credenciales.
- NO combina la configuración si ambas `[profile foo]` `[foo]` se encuentran en el mismo archivo.
- Utiliza la configuración `[profile foo]` si ambas `[profile foo]` opciones `[foo]` se encuentran en el archivo de configuración.
- Utiliza el valor de la última configuración duplicada en el mismo archivo y perfil.
- Reconoce ambos `;` y `#` para definir un comentario.
- Reconoce `;` y define un comentario `#` en las definiciones de perfil, incluso si los caracteres están junto al corchete de cierre.
- Reconoce `;` y define un comentario únicamente `#` al establecer valores si van precedidos de espacios en blanco.
- Reconoce `;` `#` y todo el contenido siguiente al establecer valores si no van precedidos de espacios en blanco.
- Considera que las credenciales basadas en roles son las credenciales de mayor prioridad. El SDK 2.x siempre usa credenciales basadas en roles si el usuario especifica la propiedad `role_arn`

- Considera las credenciales basadas en la sesión como credenciales. `second-highest-priority` El SDK 2.x siempre usa credenciales basadas en sesiones si no se usaron credenciales basadas en roles y el usuario especifica las propiedades `aws_access_key_id` `aws_session_token`
- Usa credenciales básicas si no se usan credenciales basadas en roles y sesiones y el usuario especificó la propiedad. `aws_access_key_id`

Cambios en las variables de entorno y las propiedades del sistema

1.x Variable de entorno	1.x Propiedad del sistema	2.x Variable de entorno	2.x Propiedad del sistema
<code>AWS_ACCESS_KEY_ID</code> <code>AWS_ACCESS_KEY</code>	<code>aws.accessKeyId</code>	<code>AWS_ACCESS_KEY_ID</code>	<code>aws.accessKeyId</code>
<code>AWS_SECRET_KEY</code> <code>AWS_SECRET_ACCESS_KEY</code>	<code>aws.secretKey</code>	<code>AWS_SECRET_ACCESS_KEY</code>	<code>aws.secretAccessKey</code>
<code>AWS_SESSION_TOKEN</code>	<code>aws.sessionToken</code>	<code>AWS_SESSION_TOKEN</code>	<code>aws.sessionToken</code>
<code>AWS_REGION</code>	<code>aws.region</code>	<code>AWS_REGION</code>	<code>aws.region</code>
<code>AWS_CONFIG_FILE</code>		<code>AWS_CONFIG_FILE</code>	<code>aws.configFile</code>
<code>AWS_CREDENTIALS_PROFILE_FILE</code>		<code>AWS_SHARED_CREDENTIALS_FILE</code>	<code>aws.sharedCredentialsFile</code>
<code>AWS_PROFILE</code>	<code>aws.profile</code>	<code>AWS_PROFILE</code>	<code>aws.profile</code>
<code>AWS_EC2_METADATA_DISABLED</code>	<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>AWS_EC2_METADATA_DISABLED</code>	<code>aws.disableEc2Metadata</code>

1.x Variable de entorno	1.x Propiedad del sistema	2.x Variable de entorno	2.x Propiedad del sistema
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>AWS_EC2_METADATA_SERVICE_ENDPOINT</code>	<code>aws.ec2MetadataServiceEndpoint</code>
<code>AWS_CONTAINER_CREDENTIALS_RELATIVE_URI</code>		<code>AWS_CONTAINER_CREDENTIALS_RELATIVE_URI</code>	<code>aws.containerCredentialsPath</code>
<code>AWS_CONTAINER_CREDENTIALS_FULL_URI</code>		<code>AWS_CONTAINER_CREDENTIALS_FULL_URI</code>	<code>aws.containerCredentialsFullUri</code>
<code>AWS_CONTAINER_AUTHORIZATION_TOKEN</code>		<code>AWS_CONTAINER_AUTHORIZATION_TOKEN</code>	<code>aws.containerAuthorizationToken</code>
<code>AWS_CBOR_DISABLED</code>	<code>com.amazonaws.sdk.disableCbor</code>	<code>CBOR_ENABLED</code>	<code>aws.cborEnabled</code>
<code>AWS_ION_BINARY_DISABLE</code>	<code>com.amazonaws.sdk.disableIonBinary</code>	<code>BINARY_ION_ENABLED</code>	<code>aws.binaryIonEnabled</code>
<code>AWS_EXECUTION_ENV</code>		<code>AWS_EXECUTION_ENV</code>	<code>aws.executionEnvironment</code>

1.x Variable de entorno	1.x Propiedad del sistema	2.x Variable de entorno	2.x Propiedad del sistema
	<code>com.amazonaws.sdk.disableCertificateChecking</code>	No compatible (función de solicitud)	No se admite (función de solicitud)
	<code>com.amazonaws.sdk.enableDefaultMetrics</code>	No admitido	No admitido
	<code>com.amazonaws.sdk.enableThrottledRetry</code>	No admitido	No admitido
	<code>com.amazonaws.regions.RegionUtils.fileOverride</code>	No se admite (función de solicitud)	No se admite (función de solicitud)
	<code>com.amazonaws.regions.RegionUtils.disableRemote</code>	No se admite (función de solicitud)	No se admite (función de solicitud)
	<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>	No se admite (función de solicitud)	No se admite (función de solicitud)

1.x Variable de entorno	1.x Propiedad del sistema	2.x Variable de entorno	2.x Propiedad del sistema
	com.amazonaws.sdk.enableInRegionOptimizedMode	No se admite (función de solicitud)	No se admite (función de solicitud)

Cambios en Waiters de la versión 1 a la versión 2

En este tema se detallan los cambios en la funcionalidad de Waiters desde la versión 1 (v1) a la versión 2 (v2).

En las siguientes tablas se muestra la diferencia específicamente para los camareros de DynamoDB. Los camareros de otros servicios siguen el mismo patrón.

Cambios de alto nivel

Las clases de camareros tienen el mismo artefacto de Maven que el servicio.

Cambio	v1	v2
dependencias 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></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></pre>

Cambio	v1	v2
	<pre> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifac tId>dynamodb</artif actId> </dependency> </dependencies> </pre>	<pre> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifac tId>dynamodb</artif actId> </dependency> </dependencies> </pre>
Package name	com.amazonaws.serv ices.dynamodbv2.wa iters	software.amazon.aw ssdk.services.dyna modb.waiters
Nombres de clase	AmazonDynamoDBWait ers	<ul style="list-style-type: none"> Síncrono: DynamoDbW aiter Asíncrono: DynamoDbA syncWaiter

¹ [Última versión.](#) ² [Última versión.](#)

Cambios en la API

Cambio	v1	v2
Crea un camarero	<pre> AmazonDynamoDB client = AmazonDynamoDBClie ntBuilder .standard().build(); AmazonDynamoDBWaiters waiter = client.wa iters(); </pre>	<p>Sincrónico:</p> <pre> DynamoDbClient client = DynamoDbClient.cre ate(); DynamoDbWaiter waiter = client.waiter(); </pre> <p>Asincrónico:</p>

Cambio	v1	v2
		<pre>DynamoDbAsyncClient asyncClient = DynamoDbA syncClient.create(); DynamoDbAsyncWaiter waiter = asyncClie nt.waiter();</pre>

Cambio	v1	v2
Espere hasta que exista una tabla	<p>Sincrónico:</p> <pre>waiter.tableExists() .run(new WaiterParameters<>(new DescribeTableRequest(tableName)))</pre> <p>Asincrónico:</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>Sincrónico:</p> <pre>WaiterResponse<DescribeTableResponse> waiterResponse = waiter.waitUntilTableExists(r -> r.tableName("myTable")); waiterResponse.matched().response() .ifPresent(System.out::println);</pre> <p>Asincrónico:</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>

Cambio	v1	v2
	<code>}).get();</code>	

Cambios de configuración

Cambio	v1	v2
Estrategia de sondeo (número máximo de intentos y retraso fijo)	<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>

Cambios en Amazon S3 Transfer Manager de la versión 1 a la versión 2

En este tema se detallan los cambios en Amazon S3 Transfer Manager de la versión 1 (v1) a la versión 2 (v2).

Cambios de alto nivel

Cambio	v1	v2
dependencias 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>

Cambio	v1	v2
		<pre><artifact Id>aws-crt</artifa ctId> <version> 0.28.7³</version> </dependency> </dependencies></pre>
Package name	com.amazonaws.serv ices.s3.transfer	software.amazon.aw ssdk.transfer.s3
Class name	TransferManager	S3TransferManager

¹ [Última versión.](#) ² [Última versión.](#) ³ [Última versión.](#)

Cambios de API de configuración

Opción	v1	v2
(conseguir un compilador)	<pre>TransferManagerBuilder tmBuilder = TransferManagerBui lder.standard();</pre>	<pre>S3TransferManager. Builder tmBuilder = S3TransferManager. builder();</pre>
cliente S3	<pre>tmBuilder.withS3Cl ient(...); tmBuilder.setS3C lient(...);</pre>	<pre>tmBuilder.s3Client (...);</pre>
Ejecutor	<pre>tmBuilder.withExec utorFactory(...); tmBuilder.setExecu torFactory(...);</pre>	<pre>tmBuilder.executor (...);</pre>
Cerrar los grupos de hilos	<pre>tmBuilder.withShut DownThreadPools(...);</pre>	No admitido. El ejecutor proporcionado no se apagará

Opción	v1	v2
	<pre>tmBuilder.setShutdownThreadPools(...);</pre>	cuando se cierre el <code>S3TransferManager</code> .
Tamaño mínimo de la pieza de carga	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtBuilder(). minimumPartSizeInBytes(...). build(); tmBuilder.s3Client(s3);</pre>
Umbral de la carga multiparte	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtBuilder(). thresholdInBytes(...).build(); tmBuilder.s3Client(s3);</pre>
Tamaño mínimo de la pieza de copia	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtBuilder(). minimumPartSizeInBytes(...). build(); tmBuilder.s3Client(s3);</pre>

Opción	v1	v2
Umbral de copia multiparte	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtBuilder(). threshold InBytes(...).build(); tmBuilder.s3Client(s3) ;</pre>
Inhabilitar descargas paralelas	<pre>tmBuilder.withDisableParallelDownloads(...); tmBuilder.setDisableParallelDownloads(...);</pre>	<p>Deshabilite las descargas paralelas pasando un cliente S3 estándar basado en Java al administrador de transferencias.</p> <pre>S3AsyncClient s3 = S3AsyncClient.builder().build(); tmBuilder.s3Client(s3);</pre>
Calcular siempre md5 multiparte	<pre>tmBuilder.withAlwaysCalculateMultipartMd5(...); tmBuilder.setAlwaysCalculateMultipartMd5(...);</pre>	No admitido.

Cambios de comportamiento

La transferencia paralela requiere un cliente S3 AWS basado en CRT





En la versión 2.x del SDK de Java, la característica de transferencia paralela automática (carga/descarga multiparte) está disponible a través del [cliente S3 basado en CRT de AWS](#). Para habilitar la


característica de transferencia paralela, debe agregar explícitamente la dependencia de la [biblioteca AWS Common Runtime \(CRT\)](#) para maximizar el rendimiento.

El cliente S3 AWS basado en CRT por sí solo, sin usarlo, `S3TransferManager` proporciona un rendimiento maximizado de las transferencias paralelas. `S3TransferManager` la versión 2 proporciona API adicionales que facilitan la transferencia de archivos y directorios.

La capacidad de `S3TransferManager` realizar transferencias paralelas depende de cómo `S3TransferManager` se inicie y de si la biblioteca AWS Common Runtime (CRT) se ha declarado como una dependencia.

En la siguiente tabla se describen tres escenarios de inicialización para una `S3TransferManager` versión 2 con y sin el AWS CRT declarado como dependencia.

Método de inicialización de S3 TransferManager v2	¿Se declara AWS el CRT como una dependencia?	
	yes	no
<p>Inicialice el S3TransferManager sin pasar una instancia de S3AsyncClient</p> <p>Método de creación estática:</p> <pre>S3TransferManager.create();</pre> <p>- O BIEN -</p> <p>Método de creador:</p> <pre>S3TransferManager.builder().build();</pre>	 <p>transferencia paralela automática habilitada</p>	 <p>transferencia paralela automática deshabilitada</p>
<p>Pase una instancia S3AsyncClient creada con uno de los métodos de creación crt* ()</p> <pre>S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	 <p>transferencia paralela</p>	 <p>error de tiempo de ejecución</p>

Método de inicialización de S3 TransferManager v2	¿Se declara AWS el CRT como una dependencia?	
<p>- O BIEN -</p> <pre data-bbox="115 331 1019 527">S3AsyncClient s3AsyncClient = S3AsyncClient.crtC reate(); S3TransferManager.builder().s3AsyncClient(s3A syncClient).build();</pre>	<p>automátic a habilitada</p>	
<p>Pase una instancia S3AsyncClient creada con uno de los métodos de creador estándar para que el administrador de transferencias no tenga ninguna referencia al CRT</p> <pre data-bbox="115 772 1019 968">S3AsyncClient s3AsyncClient = S3AsyncClient.buil der().build(); S3TransferManager.builder().s3AsyncClient(s3Asyn cClient).build();</pre> <p>- O BIEN -</p> <pre data-bbox="115 1079 1019 1274">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>transfere ncia paralela automática deshabilitada</p>	<div data-bbox="1338 604 1466 716" style="text-align: center;"></div> <p>transfere ncia paralela automática deshabilitada</p>

Descarga paralela mediante búsquedas por rango de bytes

Cuando la característica de transferencia paralela automática está habilitada, S3 Transfer Manager v2 utiliza recuperaciones de [rango de bytes para recuperar](#) partes específicas del objeto en paralelo (descarga multiparte). La forma en que se descarga un objeto con la versión 2 no depende de cómo se cargó originalmente. Todas las descargas pueden beneficiarse de un alto rendimiento y simultaneidad.

Por el contrario, con S3 Transfer Manager v1, importa cómo se cargó originalmente el objeto. El S3 Transfer Manager v1 recupera las partes del objeto de la misma forma en que se cargaron las partes. Si un objeto se cargó originalmente como un objeto único, la versión 1 de S3 Transfer Manager no puede acelerar el proceso de descarga mediante solicitudes secundarias.

Comportamiento de error

Con S3 Transfer Manager v1, se produce un error en una solicitud de transferencia de directorio si se produce un error en alguna solicitud secundaria. A diferencia de la v1, el future devuelto por S3 Transfer Manager v2 se completa correctamente incluso si algunas subsolicitudes fallan.

Como resultado, debe comprobar si hay errores en la respuesta utilizando el método [CompletedDirectoryDownload.failedTransfers\(\)](#) o el [CompletedDirectoryUpload.failedTransfers\(\)](#) aunque el future se complete correctamente.

Cambios en la utilidad de metadatos de EC2 de la versión 1 a versión 2

En este tema se detallan los cambios en la utilidad de metadatos Amazon Elastic Compute Cloud (EC2) del SDK para Java de la versión 1 (v1) a la versión 2 (v2).

Cambios de alto nivel

Cambio	v1	v2
dependencias Maven	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> </pre>	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> </pre>

Cambio	v1	v2
	<pre data-bbox="597 212 1023 541"> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-co re</artifactId> </dependency> </dependencies> </pre>	<pre data-bbox="1075 212 1500 814"> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>imds</artifactId> </dependency> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>apache-client³</ artifactId> </dependency> </dependencies> </pre>
Package name	com.amazonaws.util	software.amazon.awssdk.imds
Enfoque de instanciación	<p data-bbox="597 989 1023 1073">Utilice métodos de utilidad estáticos; sin instanciación:</p> <pre data-bbox="597 1115 1023 1304"> String localHostName = EC2Metada taUtils.getLocalHo stName(); </pre>	<p data-bbox="1075 989 1500 1073">Utilice un método de fábrica estático:</p> <pre data-bbox="1075 1115 1500 1262"> Ec2MetadataClient client = Ec2Metada taClient.create(); </pre> <p data-bbox="1075 1304 1500 1388">O utilice un enfoque de creación:</p> <pre data-bbox="1075 1430 1500 1703"> Ec2MetadataClient client = Ec2Metada taClient.builder() .endpointMode(Endp ointMode.IPV6) .build(); </pre>

Cambio	v1	v2
Tipos de clientes	Métodos de utilidad únicamente y síncronos: <code>EC2MetadataUtils</code>	Síncrono: <code>Ec2MetadataClient</code> Asíncrono: <code>Ec2MetadataAsyncClient</code>

¹ [Última versión.](#) ² [Última versión.](#)

³ Observe la declaración del módulo `apache-client` para la versión 2. La versión 2 de la utilidad de metadatos EC2 requiere una implementación de la interfaz `SdkHttpClient` para el cliente de metadatos síncrono o la interfaz `SdkAsyncHttpClient` para el cliente de metadatos asíncrono. La sección [???](#) muestra la lista de clientes HTTP que puede utilizar.

Solicitar metadatos

En la versión 1, se utilizan métodos estáticos que no aceptan parámetros para solicitar metadatos para un recurso de EC2. Por el contrario, debe especificar la ruta al recurso de EC2 como parámetro en la v2. En la tabla siguiente, se muestran los diferentes enfoques.

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 las [categorías de metadatos de la instancia](#) para encontrar la ruta que necesita proporcionar para solicitar un fragmento de metadatos.

Note

Cuando utilice un cliente de metadatos de instancia en la versión 2, debe intentar usar el mismo cliente para todas las solicitudes de recuperación de metadatos.

Cambios de comportamiento

Datos JSON

En EC2, el Servicio de metadatos de instancias (IMDS) que se ejecuta localmente devuelve algunos metadatos en forma de cadenas con formato JSON. Un ejemplo de ello son los metadatos dinámicos de un [documento de identidad de instancias](#).

La API v1 contiene métodos independientes para cada elemento de metadatos de identidad de la instancia, mientras que la API v2 devuelve directamente la cadena JSON. Para trabajar con la cadena JSON, puede usar la [API de documentos](#) para analizar la respuesta y navegar por la estructura JSON.

En la siguiente tabla, se compara la forma de recuperar los metadatos de un documento de identidad de instancia en las versiones 1 y 2.

Caso de uso	v1	v2
Recuperar la región	<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>
Recupera el identificador de la instancia	<pre>InstanceInfo instanceInfo =</pre>	<pre>Ec2MetadataResponse response =</pre>

Caso de uso	v1	v2
	<pre>EC2MetadataUtils.getInstanceInfo(); String instanceId = instanceInfo.getInstanceId;</pre>	<pre>client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceId = instanceInfo.asMap().get("instanceId").asString();</pre>
Recupera el tipo de la instancia	<pre>InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo(); String instanceType = instanceInfo.getInstanceType();</pre>	<pre>Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceType = instanceInfo.asMap().get("instanceType").asString();</pre>

Diferencias en la resolución de punto de conexión

La siguiente tabla muestra las ubicaciones que el SDK comprueba para resolver el punto de conexión en el IMDS. Las ubicaciones se muestran en orden de prioridad descendente.

v1	v2
Propiedad del sistema: <code>com.amazonaws.sdk.ec2MetadataService.EndpointOverride</code>	Método de configuración del creador de clientes: <code>endpoint(...)</code>
Variable de entorno: <code>AWS_EC2_METADATA_SERVICE_ENDPOINT</code>	Propiedad del sistema: <code>aws.ec2MetadataServiceEndpoint</code>

v1	v2
Valor predeterminado: <code>http://169.254.169.254</code>	Archivo de configuración: <code>~.aws/config</code> con la configuración <code>ec2_metadata_service_endpoint</code>
	Valor asociado a <code>endpoint-mode</code> resuelto
	Valor predeterminado: <code>http://169.254.169.254</code>

Resolución del punto de conexión en la versión 2

Cuando se establece explícitamente un punto de conexión mediante el creador, el valor de ese punto de conexión tiene prioridad sobre todos los demás ajustes. Cuando se ejecuta el siguiente código, la propiedad del sistema `aws.ec2MetadataServiceEndpoint` y el ajuste `ec2_metadata_service_endpoint` del archivo de configuración se ignoran si existen.

```
Ec2MetadataClient client = Ec2MetadataClient
    .builder()
    .endpoint(URI.create("endpoint.to.use"))
    .build();
```

Modo de punto de conexión

Con la versión 2, puede especificar un modo de punto de conexión para configurar el cliente de metadatos de modo que utilice los valores de punto de conexión predeterminados para IPv4 o IPv6. El modo de punto de conexión no está disponible para la versión 1. El valor predeterminado utilizado para IPv4 es `http://169.254.169.254` y `http://[fd00:ec2::254]` para IPv6.

En la siguiente tabla se muestran las distintas formas de configurar el modo de punto de conexión en orden descendente de prioridad.

		Valores posibles
Método de configuración del creador de clientes: <code>endpointMode(...)</code>	<pre>Ec2MetadataClient client = Ec2Metada taClient .builder()</pre>	<code>EndpointMode.IPV4</code> , <code>EndpointMode.IPV6</code>

		Valores posibles
	<pre>.endpointMode(EndpointMode.IPV4) .build();</pre>	
Propiedad del sistema	<code>aws.ec2MetadataServiceEndpointMode</code>	IPv4, IPv6 (no distingue entre mayúsculas y minúsculas)
Archivo de configuración: ~.aws/config	Ajuste <code>ec2_metadata_service_endpoint</code>	IPv4, IPv6 (no distingue entre mayúsculas y minúsculas)
No especificado de las formas anteriores	Se utiliza IPv4	

Cómo resuelve el SDK **endpoint** o **endpoint-mode** en la versión 2

1. El SDK usa el valor que se establece en el código del creador de clientes e ignora cualquier configuración externa. Como el SDK genera una excepción si ambos `endpoint` y `endpointMode` se invocan en el creador de clientes, el SDK usa el valor de punto de conexión del método que utilice.
2. Si no establece un valor en el código, el SDK busca en la configuración externa primero las propiedades del sistema y, después, una configuración en el archivo de configuración.
 - a. El SDK primero comprueba si hay un valor de punto de conexión. Si se encuentra un valor, se usa.
 - b. Si el SDK sigue sin encontrar ningún valor, busca la configuración del modo de punto de conexión.
3. Por último, si el SDK no encuentra ninguna configuración externa y no ha configurado el cliente de metadatos en código, el SDK utiliza el valor IPv4 de `http://169.254.169.254`.

IMDSv2

Amazon EC2 define dos enfoques para acceder a los metadatos de las instancias:

- Servicio de metadatos de instancia, versión 1 (IMDSv1): enfoque de solicitud y respuesta

- Servicio de metadatos de instancia, versión 2 (IMDSv2): enfoque orientado a la sesión

En la siguiente tabla se compara el funcionamiento de los SDK de Java con el IMDS.

v1	v2
IMDSv2 se usa de forma predeterminada	Siempre usa IMDSv2
Intenta obtener un token de sesión para cada solicitud y recurre a IMDSv1 si no logra obtener un token de sesión	Mantiene un token de sesión en una caché interna que se reutiliza para múltiples solicitudes

El SDK para Java 2.x solo admite IMDSv2 y no recurre a IMDSv1.

Diferencias de configuración

En la tabla siguiente se muestran las diferentes opciones de configuración.

Configuración	v1	v2
Reintentos	La configuración no está disponible	Configurable mediante el método de creador <code>retryPolicy(...)</code>
HTTP	El tiempo de espera de la conexión se puede configurar mediante la variable de entorno <code>AWS_METADATA_SERVICE_TIMEOUT</code> . El valor predeterminado es de 1 segundo.	La configuración está disponible pasando un cliente HTTP al método de creador <code>httpClient(...)</code> . El tiempo de espera de conexión predeterminado para los clientes HTTP es de 2 segundos.

Ejemplo de configuración HTTP v2

En el siguiente ejemplo, se muestra cómo se puede configurar el cliente de metadatos. En este ejemplo se configura el tiempo de espera de la conexión y se utiliza el cliente HTTP Apache.

```

SdkHttpClient httpClient = ApacheHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(1))
    .build();

Ec2MetadataClient imdsClient = Ec2MetadataClient.builder()
    .httpClient(httpClient)
    .build();

```

Cambios en la CloudFront presignación de Amazon de la versión 1 a la versión 2

En este tema se detallan los cambios en Amazon CloudFront desde la versión 1 (v1) a la versión 2 (v2).

Cambios de alto nivel

Cambio	v1	v2
dependencias 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> </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> </pre>

Cambio	v1	v2
	<pre> <groupId> com.amazonaws</gro upId> <artifact Id>cloudfront</art ifactId> </dependency> </dependencies> </pre>	<pre> <artifact Id>cloudfront</art ifactId> </dependency> </dependencies> </pre>
Package name	com.amazonaws.serv ices.cloudfront	software.amazon.aw ssdk.services.clou dfront
Nombres de clase	CloudFrontUrlSigner CloudFrontCookieSigner	CloudFrontUtilities SignedUrl CannedSignerRequest CustomSignerRequest

¹ [Última versión.](#) ² [Última versión.](#)

Cambios en la API

Comportamiento	v1	v2
Cree una solicitud predefinida	Los argumentos se pasan directamente a la API.	<pre> CannedSignerRequest cannedRequest = CannedSig nerRequest.builder() .resourceUrl(resou rceUrl) .privateKey(privat eKey) .keyPairId(keyPairId) </pre>

Comportamiento	v1	v2
		<pre>.expirationDate(expirationDate) .build();</pre>
Crea una solicitud personalizada	Los argumentos se pasan directamente a la API.	<pre>CustomSignerRequest customRequest = CustomSignerRequest.builder() .resourceUrl(resourceUrl) .privateKey(keyFile) .keyPairId(keyPairId) .expirationDate(expirationDate) .activeDate(activeDate) .ipRange(ipRange) .build();</pre>

Comportamiento	v1	v2
Genera una URL firmada (predefinida)	<pre>String signedUrl = CloudFrontUrlSigner.getSignedURLWith CannedPolicy(resourceUrl, keyPairId, privateKey, expirationDate);</pre>	<pre>CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create(); SignedUrl signedUrl = cloudFrontUtilities.getSignedUrlWith CannedPolicy(cannedRequest); String url = signedUrl.url();</pre>
Genera una cookie firmada (personalizada)	<pre>CookiesForCustomPolicy cookies = CloudFrontCookieSigner.getCookiesFor CustomPolicy(resourceUrl, privateKey, keyPairId, expirationDate, activeDate, ipRange);</pre>	<pre>CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create(); CookiesForCustomPolicy cookies = cloudFrontUtilities.getCookiesForCustomPolicy(customRequest);</pre>

Se refactorizaron los encabezados de las cookies en la versión 2

En Java v1, el SDK de Java proporciona los encabezados de las cookies como `Map.Entry<String, String>`

```
Map.Entry<String, String> signatureMap = cookies.getSignature();
String signatureKey = signatureMap.getKey(); // "CloudFront-Signature"
String signatureValue = signatureMap.getValue(); // "[SIGNATURE_VALUE]"
```

El SDK de Java v2 ofrece el encabezado completo como un solo `String`.

```
String signatureHeaderValue = cookies.signatureHeaderValue(); // "CloudFront-
Signature=[SIGNATURE_VALUE]"
```

Cambios en el análisis de los URI de Amazon S3 de la versión 1 a la versión 2

En este tema se detallan los cambios en el análisis de los URI de Amazon S3 de la versión 1 (v1) a la versión 2 (v2.).

Cambios de alto nivel

Para empezar a analizar un URI de S3 en la versión 1, debe crear una instancia mediante un `AmazonS3URI` constructor. En la versión 2, se `parseUri()` invoca una instancia de `S3Utilities`, para devolver un `S3URI`

Cambio	v1	v2
dependencias 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></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></pre>

Cambio	v1	v2
	<pre><artifact Id>s3</artifactId> </dependency> </dependencies></pre>	<pre><artifact Id>s3</artifactId> </dependency> </dependencies></pre>
Package name	com.amazonaws.serv ices.s3	software.amazon.aw ssdk.services.s3
Nombres de clase	AmazonS3URI	S3URI

¹ [Última versión.](#) ² [Última versión.](#)

Cambios en la API

Comportamiento	v1	v2
Analiza un URI de S3.	<pre>URI uri = URI.creat e("https://s3.amazon aws.com"); AmazonS3Uri s3Uri = new AmazonS3U RI(uri, false);</pre>	<pre>S3Client s3Client = S3Client.create(); S3Utilities s3Utiliti es = s3Client.utilities (); S3Uri s3Uri = s3Utilities.parseU ri(uri);</pre>
Recupera el nombre del bucket de un URI de S3.	<pre>String bucket = s3Uri.getBucket();</pre>	<pre>Optional<String> bucket = s3Uri.bucket();</pre>
Recupera la clave.	<pre>String key = s3Uri.get Key();</pre>	<pre>Optional<String> key = s3Uri.key();</pre>
Recupera la región.	<pre>String region = s3Uri.getRegion();</pre>	<pre>Optional<Region> region = s3Uri.region();</pre>

Comportamiento	v1	v2
		<pre>String region; if (s3Uri.region().isPresent()) { region = s3Uri.region().get().id(); }</pre>
Recupera si el URI de S3 es de estilo de ruta.	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>
Recupera el ID de la versión.	<pre>String versionId = s3Uri.getVersionId();</pre>	<pre>Optional<String> versionId = s3Uri.firstMatchingRawQueryParameter("versionId");</pre>
Recupera los parámetros de la consulta.	N/A	<pre>Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();</pre>

Cambios de comportamiento

Codificación de URL

La versión 1 ofrece la opción de pasar una marca para especificar si el URI debe estar codificado como URL. El valor predeterminado es `true`.

En la versión 2, no se admite la codificación de URL. Si trabaja con claves de objeto o parámetros de consulta que contienen caracteres reservados o no seguros, debe codificarlos mediante una URL. Por ejemplo, debe reemplazar un espacio en blanco " " por. `%20`

Cambios en la API de IAM Policy Builder de la versión 1 a la versión 2

En este tema se detallan los cambios en la API de IAM Policy Builder de la versión 1 (v1) a la versión 2 (v2).

Cambios de alto nivel

Cambio	v1	v2
dependencias 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>aws-java-sdk-co re</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>iam-policy-buil der</artifactId> </dependency> </dependencies> </pre>
Package name	com.amazonaws.auth.policy	software.amazon.awssdk.policybuilder.iam
Nombres de clase	Política Instrucción	IamPolicy IamStatement

Cambio	v1	v2
	<ul style="list-style-type: none"> • Statement.Effect • IdentityManagementActions • Resource • Entidad principal • Condición 	<ul style="list-style-type: none"> • IamEffect • IamAction • IamResource • IamPrincipal • IamCondition • IamConditionOperator • IamConditionKey

¹ [Última versión.](#) ² [Última versión.](#)

Cambios en la API

Opción	v1	v2
Cree una instancia de una política	<pre>Policy policy = new Policy();</pre>	<pre>IamPolicy.Builder policyBuilder = IamPolicy.builder(); ... IamPolicy policy = policyBuilder.buil d();</pre>
Establecer una identificación	<pre>policy.withtId(...); policy.setId(...);</pre>	<pre>policyBuilder.id(...);</pre>
Establecer versión	N/A: usa la versión predeterminada de 2012-10-17	<pre>policyBuilder.vers ion(...);</pre>
Crear declaración	<pre>Statement statement = new Statement (Effect.Allow) .withActi ons(...)</pre>	<pre>IamStatement statement = IamStatement.build er() .effect(I amEffect.ALLOW)</pre>

Opción	v1	v2
	<pre> .withCondi tions(...) .withId(. ..) .withPrin cipals(...) .withReso urces(...); </pre>	<pre> .actions(...) .notActio ns(...) .conditio ns(...) .sid(...) .principa ls(...) .notPrinc ipals(...) .resource s(...) .notResou rces(...) .build() </pre>
Establecer declaración	<pre> policy.withStateme nts(statement); policy.setStatements (statement); </pre>	<pre> policyBuilder.addS tatement(statement); </pre>

Diferencias en la construcción de una declaración

Acciones

v1

El SDK de la versión 1 tiene [enumtipos](#) de acciones de servicio que representan [Action](#) los elementos de una declaración de política. Los siguientes enum tipos son algunos ejemplos.

- [IdentityManagementActions](#)
- [DynamoDBv2Actions](#)
- [SQSActions](#)

En el siguiente ejemplo se muestra la SendMessage constante de SQSActions.

```
Action action = SQSActions.SendMessage;
```

No se puede especificar un [NotAction](#) elemento en una sentencia en la v1.

v2

En la versión 2, la [IamAction](#) interfaz representa todas las acciones. Para especificar un elemento de [acción específico del servicio](#), pase una cadena al create método, tal y como se muestra en el código siguiente.

```
IamAction action = IamAction.create("sqs:SendMessage");
```

Puede especificar a [NotAction](#) para una sentencia con v2, como se muestra en el código siguiente.

```
IamAction action = IamAction.create("sqs:SendMessage");  
IamStatement.builder().addNotAction(action);
```

Condiciones

v1

Para representar las condiciones de la declaración, el SDK de la versión 1 usa subclases de [Condition](#).

- [ArnCondition](#)
- [BooleanCondition](#)
- [DateCondition](#)
- [IpAddressCondition](#)
- [NumericCondition](#)
- [StringCondition](#)

Cada Condition subclase define un enum tipo de comparación para ayudar a definir la condición. Por ejemplo, a continuación se muestra una comparación de [cadenas diferentes](#) de una condición.

```
Condition condition = new StringCondition(StringComparisonType.StringNotLike, "key",  
"value");
```

v2

En la versión 2, se crea una condición para una declaración de política utilizando [IamCondition](#) y proporcionando una [IamConditionOperator](#), que contiene todos enums los tipos.

```
IamCondition condition = IamCondition.create(IamConditionOperator.STRING_NOT_LIKE,
    "key", "value");
```

Recursos

v1

El [Resource](#) elemento de una declaración de política está representado por la [Resource](#) clase del SDK. El ARN se proporciona como una cadena en el constructor. Las siguientes subclases proporcionan constructores de conveniencia.

- [S3 BucketResource](#)
- [S3 ObjectResource](#)
- [SQS QueueResource](#)

En la versión 1, puede especificar un [NotResource](#) elemento para a [Resource](#) llamando al `withIsNotType` método, tal y como se muestra en la siguiente sentencia.

```
Resource resource = new Resource("arn:aws:s3:::mybucket").withIsNotType(true);
```

v2

En la versión 2, se crea un [Resource](#) elemento pasando un ARN al `IamResource.create` método.

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");
```

[IamResource](#) Se puede establecer un [NotResource](#) elemento como elemento, tal y como se muestra en el siguiente fragmento.

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");
IamStatement.builder().addNotResource(resource);
```

`IamResource.ALL` representa todos los recursos.

Entidades principales

v1

El SDK de la versión 1 ofrece las siguientes [Principal](#) clases para representar los tipos de directores que incluyen a todos los miembros:

- `AllUsers`
- `AllServices`
- `AllWebProviders`
- `All`

No se puede añadir un [NotPrincipal](#) elemento a una declaración.

v2

En la versión 2, `IamPrincipal.ALL` representa todos los principios:

Para representar a todos los miembros de otros tipos de principios, utilice las [IamPrincipalType](#) clases al crear un `IamPrincipal`

- `IamPrincipal.create(IamPrincipalType.AWS, "*")` para todos los usuarios.
- `IamPrincipal.create(IamPrincipalType.SERVICE, "*")` para todos los servicios.
- `IamPrincipal.create(IamPrincipalType.FEDERATED, "*")` para todos los proveedores web.
- `IamPrincipal.create(IamPrincipalType.CANONICAL_USER, "*")` para todos los usuarios canónicos.

Puede utilizar el `addNotPrincipal` método para representar un [NotPrincipal](#) elemento al crear una declaración de política, como se muestra en la siguiente declaración.

```
IamPrincipal principal = IamPrincipal.create(IamPrincipalType.AWS,
    "arn:aws:iam::444455556666:root");
IamStatement.builder().addNotPrincipal(principal);
```

Utilizar el SDK para Java 1.x y 2.x en paralelo

Puede utilizar ambas versiones de AWS SDK for Java en sus proyectos.

A continuación, se muestra un ejemplo del archivo `pom.xml` para un proyecto que utiliza Amazon S3 de la versión 1.11.x y DynamoDB de la versión 2.16.1.

Example Ejemplo de POM

En este ejemplo se muestra una entrada de archivo `pom.xml` para un proyecto que utiliza las versiones 1.x y 2.x del 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>
```

Clave OpenPGP para AWS SDK for Java

Todos los artefactos de Maven disponibles públicamente para AWS SDK for Java están firmados con el estándar OpenPGP. La clave pública que necesita para verificar la firma de un artefacto está disponible en la siguiente sección.

Clave actual

En la siguiente tabla se muestra la información clave de OpenPGP para las versiones actuales del SDK para Java 1x y del SDK para Java 2.x.

ID de clave	0xAC107B386692DADD
Tipo	RSA
Tamaño	4096/4096
Creado	30-06-2016
Expires	2024-10-08
ID de usuario	SDK de AWS y herramientas < aws-dr-tools@amazon.com >
Huella digital clave	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

Para copiar la siguiente clave pública de OpenPGP para el SDK para Java en el portapapeles, seleccione el icono “Copiar” en la esquina superior derecha.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
xsFNBFd1gAUBEACqbmFbxdJgz1lD7wr1skQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYp0viSWsX2psgvdmeyUpW9ap01rThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKR0mRODzUuaokLPo24pfm9bnr1RnRtw5ktPAA5bM9ZZaGKriej
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
pUwgp0MTo25gWxkvJlSJKU0b6b1786WnySIzF2gxqlkkEmB14RAssQkeXjrSmGws
MDyHNqyJeYFus18sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MY1nasd4bW2RK1sr7plkBf8QRe6biiQRf3KD
0Sn5CbmXpAcHJ1ZHzRRdkXZDNQC6vCjXsy1300TrhJtAV1Yq347uyUbVi291ISVg
roUVtprismHoEk5Go0THbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbtnbs
/Hd981FdVghYYvq//gTakJk0WxfGq030wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsF1BBgBCgAPBQJXdYAFaHsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0H0VHwQsaW
jMrGj000MFzxGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+Yn0Qf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
l6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB1kke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeq167ggx+wFjKDcfhZA7m70LH0DysrGVCLcmuinUBaN1HmLDcGY
XZ+kMCoXf0bpuCVByQmNJgEb47EIFlx/+TEeNHKM0+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv00PCv0
WTA/MClv28y0PrSvcmXnduLtkBEX7TISMPW+n+0Ta63/z4YFFEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```

Historial de documentos

En este tema se describen los cambios importantes que se han AWS SDK for Java introducido en la Guía para desarrolladores a lo largo de su historia.

Esta guía se publicó por última vez el 18 de junio de 2024.

Cambio	Descripción	Fecha
the section called “Reintentos”	Añadir tema de estrategia de reintento	18 de junio de 2024
Cómo configurar el TTL de JVM	Elimine las instrucciones para establecer una propiedad <code>networkaddress.cache.ttl</code> de seguridad mediante una propiedad del sistema de línea de comandos de Java.	21 de mayo de 2024
the section called “Reduzca el tiempo de inicio del SDK para AWS Lambda”	Actualice la recomendación del cliente HTTP para reducir el tiempo de inicio de AWS Lambda	14 de mayo de 2024
the section called “Métricas de los clientes de servicio”	Reorganice los elementos de la tabla de métricas	1 de mayo de 2024
the section called “Resolución de problemas”	Agregue un tema de solución de problemas.	26 de abril de 2024
the section called “Métricas recopiladas con cada solicitud”	Agrega nuevas métricas reportadas por el SDK.	26 de abril de 2024
the section called “Configurar el TTL de JVM para las búsquedas de nombres DNS”	Cambia el TTL de búsqueda de DNS recomendado a 5 segundos.	23 de abril de 2024

Cambio	Descripción	Fecha
the section called “Nombre del paquete a las asignaciones de ArtifactsID”	Agregue el nombre del paquete al tema de mapeo ArtifactsID de Maven.	17 de abril de 2024
the section called “Usar las métricas del SDK”	Agregue detalles de configuración a la sección de métricas.	12 de abril de 2024
the section called “API del creador de políticas de IAM”	Añada la información de migración de la API de IAM Policy Builder.	11 de abril de 2024
???	Actualice la información del proxy HTTP.	3 de abril de 2024
the section called “De forma segura”	Agregue instrucciones para deshabilitar IMDSv1.	14 de marzo de 2024
the section called “tep-by-step Instrucciones S”	Agregue instrucciones de step-by-step migración.	8 de marzo de 2024
Migrar a la versión 2	Actualice el tema de migración .	14 de febrero de 2024
the section called “Configurar clientes AWS HTTP basados en CRT”	Agregue información sobre el cliente HTTP síncrono AWS basado en CRT.	5 de enero de 2024
the section called “Amazon Cognito Identity” y the section called “Amazon Cognito Identity Provider”	Los ejemplos de Amazon Cognito se han trasladado a la sección de ejemplos de código.	28 de diciembre de 2023
Uso de características del SDK	Se ha rediseñado el tema de las características del SDK.	11 de diciembre de 2023
Clave OpenPGP	Proporcionar la clave OpenPGP actual.	6 de diciembre de 2023

Cambio	Descripción	Fecha
the section called “Cambios de serialización”	Describir las diferencias de serialización entre las versiones 1 y 2 del SDK para Java.	5 de diciembre de 2023
the section called “S3 Transfer Manager”	Añadida una sección que detalla los cambios en el S3 Transfer Manager de la versión 1 a la versión 2.	13 de noviembre de 2023
the section called “Referencia de anotaciones”	Añadida una lista de anotaciones de clases de datos que se puedan utilizar con el cliente mejorado de DynamoDB.	30 de octubre de 2023
???	Añadida información sobre el estado de migración de las bibliotecas y utilidades del SDK para Java v1.x a la v2.x	17 de octubre de 2023
???	Actualizado el tema de configuración de Gradle	17 de octubre de 2023
the section called “Ignorar atributos nulos de los objetos anidados”	Añadida información sobre la anotación <code>@DynamoDbIgnoreNulls</code> del cliente mejorado DynamoDB	22 de septiembre de 2023
the section called “Acceso entre regiones”	Añadida información acerca del acceso entre regiones a los buckets de Amazon S3.	31 de agosto de 2023
the section called “Conserva los objetos vacíos”	Añadida una sección que analiza la anotación <code>@DynamoDbPreserveEmptyObject</code> .	25 de agosto de 2023

Cambio	Descripción	Fecha
???	Actualizada la sección de clientes del servicio.	15 de agosto de 2023
the section called “Recomendaciones sobre clientes”	Desde la versión 0.23, AWS CRT es compatible con sistemas operativos basados en MUSL, como Alpine Linux. Las recomendaciones de los clientes HTTP ahora reflejan la compatibilidad con musl.	11 de agosto de 2023
the section called “Crear políticas de IAM”	Añadida la sección API del creador de políticas de IAM	31 de julio de 2023
the section called “Introducción”	Corregidos varios fragmentos de la sección Introducción del tema Cliente mejorado de DynamoDB	24 de julio de 2023
the section called “Configurar proxies HTTP”	Añadidos información y ejemplos de compatibilidad con el proxy HTTP para cada cliente HTTP.	2 de junio de 2023
Reorganizada la tabla de contenido	Promocionada la sección Ejemplos de código y Trabaje con Servicios de AWS a entradas del índice nivel superior.	24 de mayo de 2023
the section called “Añadir dependencia de registro”	Muestra las dependencias de Gradle en la sección de registro.	23 de mayo de 2023
the section called “Trabaje con resultados paginados”	Actualizado el tema de paginación.	18 de mayo de 2023

Cambio	Descripción	Fecha
the section called “Configurar un proyecto Gradle”	Actualizada la configuración del proyecto de Gradle.	3 de mayo de 2023
API de cliente mejorado de DynamoDB	Publicado el tema reescrito sobre la API de cliente mejorado de DynamoDB.	28 de abril de 2023
Actualizadas las instrucciones del tutorial de introducción	Modificado el arquetipo de Maven para incluir la opción CredentialsProvider; las instrucciones se modificaron en consecuencia.	11 de abril de 2023
the section called “Recomendaciones sobre clientes”	Añadida una guía para la toma de decisiones para clientes HTTP	30 de marzo de 2023
Actualizaciones de las prácticas recomendadas de IAM	Se ha actualizado la guía para implementar las prácticas recomendadas de IAM. Para obtener más información, consulte prácticas recomendadas de seguridad en IAM .	14 de marzo de 2023
the section called “Volver a cargar credenciales del perfil”	Añadida una sección sobre la recarga de las credenciales del perfil.	9 de febrero de 2023
the section called “Configurar clientes AWS HTTP basados en CRT”	Tema actualizado para el lanzamiento de GA.	8 de febrero de 2023
the section called “Trabajar con metadatos de la instancia de Amazon EC2”	Añadido un ejemplo guiado para el cliente Java SDK para el servicio de metadatos de instancias Amazon S3.	1 de febrero de 2023

Cambio	Descripción	Fecha
the section called “Utilizar un cliente S3 de alto rendimiento”	Agregue una sección para el cliente S3 basado en CRT AWS .	19 de diciembre de 2022
the section called “Transferir archivos y directorios”	Actualizados los ejemplos del Amazon S3 Transfer Manager para la versión de GA.	19 de diciembre de 2022
the section called “Prácticas recomendadas”	Añadida la sección de prácticas recomendadas.	18 de noviembre de 2022
the section called “Cargar las credenciales temporales de un proceso externo”	Añadida una sección sobre la carga de credenciales de un proceso externo.	15 de noviembre de 2022
the section called “Métricas de los clientes de servicio”	Actualizado el listado de métricas con el requisito de uso de clientes HTTP.	9 de noviembre de 2022
the section called “Transferir archivos y directorios”	Código de ejemplo corregido.	2 de noviembre de 2022
the section called “Reduzca el tiempo de inicio del SDK para AWS Lambda”	Sección actualizada con opciones adicionales para reducir el tiempo de arranque de Lambda.	1 de noviembre de 2022
the section called “Clientes de HTTP”	Añadida información de configuración para cubrir todos los clientes HTTP del SDK.	26 de octubre de 2022
the section called “Registro”	Tema de registro actualizado para incluir detalles de registro de red para todos los clientes HTTP.	4 de octubre de 2022

Cambio	Descripción	Fecha
the section called “AWS servicios de bases de datos”	Se agregó una sección de información general sobre los servicios de AWS bases de datos y el SDK for Java 2.x.	13 de septiembre de 2022
Se retira EC2-Classic Networking	EC2-Classic se retirará el 15 de agosto de 2022.	28 de julio de 2022
the section called “Opciones de autenticación adicionales”	Actualización de la dependencia necesaria para la autenticación de inicio de sesión único.	18 de julio de 2022
the section called “seguridad de la capa de transporte (TLS)”	Información de seguridad TLS actualizada.	8 de abril de 2022
the section called “Opciones de autenticación adicionales”	Se ha añadido más información sobre la configuración y el uso de credenciales.	22 de febrero de 2021
the section called “Configurar un proyecto de imagen nativa de GraalVM”	Nuevo tema para configurar un proyecto de GraalVM Native Image.	18 de febrero de 2021
the section called “Sondear los estados de recursos”	Lanzamiento de esperadores; se ha añadido un tema para la nueva característica.	30 de septiembre de 2020
the section called “Usar las métricas del SDK”	Métricas publicadas; tema añadido para la nueva característica.	17 de agosto de 2020
the section called “Amazon SNS”	Se agregaron temas de ejemplo para Amazon SNS	30 de mayo de 2020

Cambio	Descripción	Fecha
the section called “Reduzca el tiempo de inicio del SDK para AWS Lambda”	Se agregó un tema sobre el rendimiento de las AWS Lambda funciones.	29 mayo de 2020
the section called “Configurar el TTL de JVM para las búsquedas de nombres DNS”	Añadido el tema JVM TTL DNS caching.	27 de abril de 2020
the section called “Configurar un proyecto Apache Maven”, the section called “Configurar un proyecto Gradle”	Nuevos temas de configuración de Maven y Gradle.	21 de abril de 2020
the section called “seguridad de la capa de transporte (TLS)”	Se ha agregado TLS 1.2 a la sección de seguridad.	19 de marzo de 2020
the section called “Suscribirse a Amazon Kinesis Data Streams”	Se han añadido ejemplos de Kinesis transmisiones.	2 de agosto de 2018
the section called “Trabaje con resultados paginados”	Se ha añadido un tema sobre la paginación automática.	5 de abril de 2018
???	Se han añadido temas de ejemplo para IAM Amazon EC2, CloudWatch y DynamoDB.	29 de diciembre de 2017
the section called “Amazon S3”	Se agregó un ejemplo de getObject para. Amazon S3	7 de agosto de 2017
the section called “Usar la programación asíncrona”	Se ha añadido el tema sobre métodos asíncronos.	4 de agosto de 2017
Publicación GA del AWS SDK for Java 2.x	AWS SDK for Java Publicada la versión 2 (v2).	28 de junio de 2017

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.