Guía para desarrolladores

AWS SDK para Kotlin

AWS SDK para Kotlin: Guía para desarrolladores

Table of Contents

¿Qué es el AWS SDK para Kotlin?

AWS SDK para Kotlin Proporciona Kotlin APIs para Amazon Web Services. Con élSDK, puede crear aplicaciones de Kotlin que funcionen con Amazon S3, AmazonEC2, Amazon DynamoDB y más. Con KotlinSDK, puedes dirigirte a la JVM plataforma o al API nivel 24 de Android o superior. Support para plataformas adicionales como JavaScript Native estará disponible en futuras versiones.

Para hacer un seguimiento de las próximas funciones de las próximas versiones, consulta nuestra hoja de ruta en GitHub.

Comenzar a utilizar AWS SDK for JavaScript

Para empezar con ellasSDK, sigue el Introducción tutorial.

Para configurar su entorno de desarrollo, consulte Configuración.

Para crear y configurar clientes de servicio a los que realizar solicitudes Servicios de AWS, consulte Configuración. Para obtener información sobre las diversas funciones delSDK, consulteUtilizar SDK.

Para ver casos de uso y ejemplos de cómo realizar API operaciones específicas, consulteEjemplos de código.

Mantenimiento y soporte para SDK las versiones principales

Para obtener información sobre el mantenimiento y el soporte de las versiones SDK principales y sus dependencias subyacentes, consulte los siguientes temas de la Guía de referencia de las herramientas AWS SDKs y herramientas:

- AWS SDKsy Política de mantenimiento de herramientas
- AWS SDKsMatriz de soporte de versiones y herramientas

Recursos adicionales

Además de esta guía, los siguientes son valiosos recursos en línea SDK para los desarrolladores de Kotlin:

- AWS blog para desarrolladores

- Foros de desarrolladores
- SDKfuente (GitHub)
- AWS Code Sample Catalog
- @awsdevelopers (X, anteriormente Twitter)

Comience con el SDK para Kotlin

AWS SDK para Kotlin Proporciona Kotlin APIs para cada uno Servicio de AWS. Con élSDK, puede crear aplicaciones de Kotlin que funcionen con Amazon S3, AmazonEC2, Amazon DynamoDB y más.

En este tutorial, se muestra cómo usar Gradle para definir las dependencias de. AWS SDK para Kotlin A continuación, se crea un código que escribe datos en una tabla de DynamoDB. Si bien es posible que desee utilizar las funciones de unaIDE, todo lo que necesita para este tutorial es una ventana de terminal y un editor de texto.

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 comenzar este tutorial, necesita un conjunto de permisos de IAM Identity Center que pueda acceder a DynamoDB y necesita un entorno de desarrollo de Kotlin configurado IAM con la configuración de inicio de sesión único de Identity Center al que acceder. AWS

Siga las instrucciones Configuración básica de esta guía para obtener la configuración básica de este tutorial.

Una vez que hayas configurado tu entorno de desarrollo con un acceso de inicio de sesión único para Kotlin SDK y tengas una sesión activa en el portal de AWS acceso, continúa con el paso 2.

Paso 2: Crear el proyecto

Para crear el proyecto de este tutorial, primero usa Gradle para crear los archivos básicos de un proyecto de Kotlin. Luego, actualiza los archivos con la configuración, las dependencias y el código necesarios para el. AWS SDK para Kotlin

Para crear un proyecto nuevo con Gradle

> **Note**
>
> En este tutorial, se usa la versión 8.11.1 de Gradle con el `gradle init` comando, que ofrece cinco instrucciones en el paso 3 que se muestra a continuación. Si utilizas una versión diferente de Gradle, es posible que las instrucciones difieran, al igual que las versiones preconfiguradas de los artefactos.

1. Crea un nuevo directorio llamado `getstarted` en la ubicación que elijas, como el escritorio o la carpeta principal.
2. Abre una ventana de terminal o línea de comandos y navega hasta el `getstarted` directorio que has creado.
3. Usa el siguiente comando para crear un nuevo proyecto de Gradle y una clase básica de Kotlin.

   ```
   gradle init --type kotlin-application --dsl kotlin
   ```

   - Cuando se te pida el objetivo`Java version`, presiona `Enter` (el valor predeterminado es). `21`
   - Cuando se le pida`Project name`, presione `Enter` (el valor predeterminado es el nombre del directorio, `getstarted` en este tutorial).
   - Cuando se le pida`application structure`, presione `Enter` (el valor predeterminado es). `Single application project`
   - Cuando se le solicite`Select test framework`, pulse `Enter` (por defecto es). `kotlin.test`
   - Cuando se le pida`Generate build using new APIs and behavior`, pulse `Enter` (por defecto es). no

Para configurar su proyecto con dependencias para Amazon S3 AWS SDK para Kotlin y Amazon S3

- En el `getstarted` directorio que creó en el procedimiento anterior, sustituya el contenido del `settings.gradle.kts` archivo por el siguiente contenido, *`X.Y.Z`* sustituyéndolo SDK por la última versión de Kotlin:

  ```
  dependencyResolutionManagement {
      repositories {
          mavenCentral()
      }
  ```

```
    versionCatalogs {
        create("awssdk") {
            from("aws.sdk.kotlin:version-catalog:X.Y.Z")
        }
    }
}

plugins {
    // Apply the foojay-resolver plugin to allow automatic download of JDKs.
    id("org.gradle.toolchains.foojay-resolver-convention") version "0.8.0"
}

rootProject.name = "getstarted"
include("app")
```

- Navega hasta el `gradle` directorio que se encuentra dentro del `getstarted` directorio. Sustituya el contenido del archivo de catálogo de versiones denominado `libs.versions.toml` por el siguiente contenido:

```
[versions]
junit-jupiter-engine = "5.10.3"

[libraries]
junit-jupiter-engine = { module = "org.junit.jupiter:junit-jupiter-engine",
 version.ref = "junit-jupiter-engine" }

[plugins]
kotlin-jvm = { id = "org.jetbrains.kotlin.jvm", version = "2.1.0" }
```

- Vaya al directorio `app` y abra el archivo `build.gradle.kts`. Sustituya el contenido por el siguiente código y, a continuación, guarde los cambios.

```
plugins {
    alias(libs.plugins.kotlin.jvm)
    application
}


dependencies {
    implementation(awssdk.services.s3) // Add dependency on the AWS SDK para Kotlin's
 S3 client.
```

```
    testImplementation("org.jetbrains.kotlin:kotlin-test-junit5")
    testImplementation(libs.junit.jupiter.engine)
    testRuntimeOnly("org.junit.platform:junit-platform-launcher")
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(21)
    }
}

application {
    mainClass = "org.example.AppKt"
}

tasks.named<Test>("test") {
    useJUnitPlatform()
}
```

La `dependencies` sección contiene una `implementation` entrada para el módulo Amazon S3 del AWS SDK para Kotlin. El compilador de Gradle está configurado para usar Java 2.1 en la `java` sección.

Paso 3: Escribir el código

Una vez creado y configurado el proyecto, edita la clase predeterminada del proyecto App para usar el siguiente código de ejemplo.

1. En la carpeta del proyectoapp, navegue hasta el directorio`src/main/kotlin/org/example`. Abra el archivo `App.kt`.
2. Sustituya su contenido por el siguiente código y guarde el archivo.

```
package org.example

import aws.sdk.kotlin.services.s3.*
import aws.sdk.kotlin.services.s3.model.BucketLocationConstraint
import aws.smithy.kotlin.runtime.content.ByteStream
import kotlinx.coroutines.runBlocking
import java.util.UUID

val REGION = "us-west-2"
```

```
val BUCKET = "bucket-${UUID.randomUUID()}"
val KEY = "key"

fun main(): Unit = runBlocking {
    S3Client
        .fromEnvironment { region = REGION }
        .use { s3 ->
            setupTutorial(s3)

            println("Creating object $BUCKET/$KEY...")

            s3.putObject {
                bucket = BUCKET
                key = KEY
                body = ByteStream.fromString("Testing with the Kotlin SDK")
            }

            println("Object $BUCKET/$KEY created successfully!")

            cleanUp(s3)
        }
}

suspend fun setupTutorial(s3: S3Client) {
    println("Creating bucket $BUCKET...")
    s3.createBucket {
        bucket = BUCKET
        if (REGION != "us-east-1") { // Do not set location constraint for us-east-1.
            createBucketConfiguration {
                locationConstraint = BucketLocationConstraint.fromValue(REGION)
            }
        }
    }
    println("Bucket $BUCKET created successfully!")
}

suspend fun cleanUp(s3: S3Client) {
    println("Deleting object $BUCKET/$KEY...")
    s3.deleteObject {
        bucket = BUCKET
        key = KEY
    }
    println("Object $BUCKET/$KEY deleted successfully!")
```

```
    println("Deleting bucket $BUCKET...")
    s3.deleteBucket {
        bucket = BUCKET
    }
    println("Bucket $BUCKET deleted successfully!")
}
```

Paso 4: Compilar y ejecutar la aplicación

Una vez creado el proyecto y que contenga la clase de ejemplo, compile y ejecute la aplicación.

1. Abra una ventana de terminal o símbolo del sistema y desplácese hasta el directorio del proyecto `getstarted`.
2. Use el siguiente comando para crear y ejecutar la aplicación:

```
gradle run
```

> **Note**
>
> Si obtiene una`IdentityProviderException`, es posible que no tenga una sesión de inicio de sesión único activa. Ejecute el `aws sso login` AWS CLI comando para iniciar una nueva sesión.

La aplicación llama a la createBucketAPIoperación para crear un nuevo depósito de S3 y, a continuación, llama putObjecta colocar un objeto nuevo en el nuevo depósito de S3.

Al final de la `cleanUp()` función, la aplicación elimina el objeto y, a continuación, elimina el depósito de S3.

Para ver los resultados en la consola Amazon S3

1. DentroApp`.kt`, comente la línea `cleanUp(s3)` de la `runBlocking` sección y guarde el archivo.
2. Reconstruya el proyecto y coloque un objeto nuevo en un nuevo depósito de S3 ejecutando`gradle run`.
3. Inicie sesión en la consola de Amazon S3 para ver el nuevo objeto en el nuevo bucket de S3.

Después de ver el objeto, elimine el bucket de S3.

Success

Si tu proyecto de Gradle se creó y se ejecutó sin errores, enhorabuena. Creaste correctamente tu primera aplicación de Kotlin con. AWS SDK para Kotlin

Limpieza

Cuando haya terminado de desarrollar su nueva aplicación, elimine todos AWS los recursos que haya creado durante este tutorial para evitar incurrir en cargos. Es posible que también desee eliminar o archivar la carpeta del proyecto (`get-started`) que creó en el paso 2.

Siga estos pasos para limpiar los recursos:

- Si ha comentado la llamada a la `cleanUp()` función, elimine el bucket de S3 mediante la consola de Amazon S3.

Pasos a seguir a continuación

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

- Pasos de configuración adicionales para trabajar con Kotlin SDK
- Configuración de SDK para Kotlin
- Uso del SDK para Kotlin
- Seguridad para SDK Kotlin

Configure el AWS SDK para Kotlin

Para solicitar el Servicios de AWS uso de AWS SDK para Kotlin, necesita lo siguiente:

- La posibilidad de iniciar sesión en el portal de AWS acceso
- Permiso para usar los AWS recursos que su aplicación necesita
- 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 credenciales del Centro de IAM Identidad para que SDK pueda obtener AWS las credenciales
    - El `credentials` archivo contiene credenciales temporales
  - Una herramienta de automatización de compilaciones como Gradle o Maven
- Una sesión activa en el portal de AWS acceso cuando esté listo para ejecutar la aplicación

En este tema

Configuración básica

Descripción general

Para desarrollar correctamente aplicaciones a las que se acceda Servicios de AWS mediante el AWS SDK para Kotlin, se deben cumplir los siguientes requisitos.

- Debe poder iniciar sesión en el portal de acceso a AWS disponible en el AWS IAM Identity Center.
- Los permisos del IAM rol configurado SDK deben permitir el acceso al 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 de IAM Identity Center para que SDK puedan obtener AWS las credenciales.
  - 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 IDEAIntelliJ o Eclipse.

  Cuando utilizas unIDE, también puedes integrarlo AWS Toolkit para trabajar con él más fácilmente. Servicios de AWS Los AWS Toolkit para IntelliJy AWS Toolkit for Eclipseson dos kits de herramientas que puede utilizar.

- Una sesión activa en el portal de AWS acceso cuando esté listo para ejecutar la aplicación. La utiliza AWS Command Line Interface para iniciar el proceso de inicio de sesión en el portal de AWS acceso de IAM Identity Center.

> **Important**
>
> En las instrucciones de esta sección de configuración se presupone que usted o su organización utilizan IAM Identity Center. Si su organización utiliza un proveedor de identidad externo que funciona de forma independiente de IAM Identity Center, averigüe cómo puede obtener credenciales temporales para que las SDK utilice Kotlin. Sigue estas instrucciones para añadir credenciales temporales al `~/.aws/credentials` archivo.
> Si su proveedor de identidad agrega credenciales temporales automáticamente al `~/.aws/credentials` archivo, asegúrese de que el nombre del perfil sea `[default]` tal que no necesite proporcionar un nombre de perfil al SDK o AWS CLI.

Posibilidad de iniciar sesión en el portal de AWS acceso

El portal de AWS acceso es la ubicación web en la que se inicia sesión manualmente en el Centro de IAM identidad. El formato de URL es `d-xxxxxxxxxx.awsapps.com/start` o*`your_subdomain`*`.awsapps.com/start`.

Si no está familiarizado con el portal de AWS acceso, siga las instrucciones sobre el acceso a las cuentas que figuran en el tema de autenticación del Centro de IAM Identidad de la Guía de referencia sobre herramientas AWS SDKs y herramientas.

Configure el acceso de inicio de sesión único para SDK

Tras completar el paso 2 de la sección de acceso mediante programación para poder utilizar la SDK autenticación de 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 de Kotlin utiliza la configuración del proveedor de SSO tokens del perfil para adquirir las credenciales antes de enviar las solicitudes a AWS. El `sso_role_name` valor, que es un IAM rol conectado a un conjunto de permisos del Centro de IAM Identidad, debería permitir el acceso al que Servicios de AWS se usa en tu aplicación.

  El siguiente `config` archivo de ejemplo muestra un perfil predeterminado configurado con la configuración de un proveedor de SSO token. La configuración `sso_session` del perfil hace referencia a la sección `sso-session` nombrada. La `sso-session` sección contiene los ajustes 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 obtener más información sobre los ajustes utilizados en la configuración del proveedor de SSO token, consulte la configuración del proveedor de SSO token en la Guía de referencia de herramientas AWS SDKs y herramientas.

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 SDKs referencia.

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 poder utilizar la SDK autenticación de 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 `--profile` opción. Si la configuración de tu proveedor de SSO token usa un perfil con nombre, el comando es`aws sso login --profile named-profile`.

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

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

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

> **Note**
>
> Si ya tiene una sesión activa en el portal de acceso a AWS y 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 .
> Selecciona Permitir para autorizar el acceso a tu información para AWS CLI y SDK para Kotlin.

Instalar Java y una herramienta de compilación

Su entorno de desarrollo debe contar con lo siguiente:

- JDK8 o una versión posterior. AWS SDK para Kotlin Funciona con el kit de desarrollo Java SE de Oracle y con distribuciones del kit de desarrollo Open Java (OpenJDK) Amazon Corretto, como Red Hat Open JDK y AdoptOpenJDK.
- Una herramienta de compilación o IDE compatible con Maven Central, como Apache Maven, Gradle o IntelliJ.
  - Para obtener información sobre cómo instalar y utilizar Maven, consulte http://maven.apache.org/.
  - Para obtener información sobre cómo instalar y usar Gradle, consulte https://gradle.org/.
  - Para obtener información sobre cómo instalar y usar IDEA IntelliJ, consulte. https://www.jetbrains.com/idea/

Utilizar credenciales temporales

Como alternativa a configurar el acceso de inicio de sesión único a IAM Identity Center para elSDK, puede configurar su entorno de desarrollo con credenciales temporales.

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 usa SDK para Kotlin si no se especifica un perfil con nombre específico.
4. Inicia sesión en el portal de AWS acceso
5. Siga estas instrucciones que aparecen en el apartado Actualización manual de credenciales para copiar las credenciales de los IAM roles del portal de AWS acceso.

a. En el paso 4 de las instrucciones enlazadas, elija el nombre del IAM rol que concede el acceso para sus necesidades de desarrollo. Este rol suele tener un nombre similar al PowerUserAccessde Desarrollador.

b. Para el paso 7, seleccione la opción Agregar manualmente un perfil a su archivo de credenciales de AWS y copie el contenido.

6. Pegue las credenciales copiadas en su archivo `credentials` local y elimine el nombre del perfil generado. Su archivo debería parecerse a lo siguiente:

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token=IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONG
```

7. Guarde el archivo `credentials`.

El SDK de Kotlin accederá a estas credenciales temporales cuando cree un cliente de servicio y las utilice para cada solicitud. La configuración del IAM rol elegido en el paso 5a determina durante cuánto tiempo son válidas las credenciales temporales. La duración máxima es de doce horas.

Cuando las credenciales temporales caduquen, repita los pasos del 4 al 7.

Cree archivos de compilación del proyecto

Después de configurar el acceso de inicio de sesión único y tu entorno de desarrollo, crea un proyecto de Kotlin con la herramienta de compilación que prefieras. En el archivo de compilación, especifica las dependencias a las Servicios de AWS que debe acceder tu aplicación.

Para ver la lista de todos los posibles nombres de artefactos de Maven, consulta la API documentación de referencia. Para encontrar la versión más reciente deSDK, consulta la última versión en. GitHub

Los siguientes archivos de compilación de ejemplo proporcionan los elementos necesarios para empezar a codificar un proyecto con siete Servicios de AWS.

Gradle (literal notation)

El siguiente `build.gradle.kts` archivo de ejemplo tiene dependencias para siete Servicios de AWS. Puede navegar hasta el *`X.Y.Z`* enlace para ver la última versión disponible.

```
plugins {
    kotlin("jvm") version "X.Y.Z"
    application
}

group = "org.example"
version = "1.0-SNAPSHOT"

repositories {
    mavenCentral()
}

dependencies {
// Use bill of materials (BOM) to get recommended Kotlin SDK dependency versions.
    implementation(platform("aws.sdk.kotlin:bom:X.Y.Z"))
    implementation(platform("org.apache.logging.log4j:log4j-bom:X.Y.Z"))

    implementation("aws.sdk.kotlin:s3")
    implementation("aws.sdk.kotlin:dynamodb")
    implementation("aws.sdk.kotlin:iam")
    implementation("aws.sdk.kotlin:cloudwatch")
    implementation("aws.sdk.kotlin:cognitoidentityprovider")
    implementation("aws.sdk.kotlin:sns")
    implementation("aws.sdk.kotlin:pinpoint")
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")

    // Test dependency.
    testImplementation(kotlin("test"))
}

tasks.test {
    useJUnitPlatform()
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(X*)
    }
}

application {
    mainClass = "org.example.AppKt"
}
```

* Versión Java, por ejemplo 17 o21.

El `settings.gradle.kts` archivo contiene la configuración general del proyecto.

```
plugins {
    id("org.gradle.toolchains.foojay-resolver-convention") version "X.Y.Z"
}
rootProject.name = "your-project-name"
```

Gradle (version catalog)

AWS SDK para Kotlin Publica un catálogo de versiones de Gradle que puede ayudarte a descubrir los nombres de las dependencias y a mantener los números de versión sincronizados en varios artefactos.

Ten en cuenta que los catálogos de versiones son una función de vista previa de Gradle antes de la versión 8. Según la versión de Gradle que utilices, es posible que tengas que activarla a través de la vista previa de funciones. API

Para usar un catálogo de versiones de Gradle

1. En tu `settings.gradle.kts` archivo, agrega un `versionCatalogs` bloque dentro del `dependencyResolutionManagement` bloque.

   El siguiente archivo de ejemplo configura el catálogo de AWS SDK para Kotlin versiones. Puede navegar hasta el *X.Y.Z* enlace para ver la última versión disponible.

   ```
   plugins {
       id("org.gradle.toolchains.foojay-resolver-convention") version "X.Y.Z"
   }
   rootProject.name = "your-project-name"

   dependencyResolutionManagement {
       repositories {
           mavenCentral()
       }

       versionCatalogs {
           create("awssdk") {
               from("aws.sdk.kotlin:version-catalog:X.Y.Z")
           }
       }
   ```

```
}
```

2. Declare las dependencias `build.gradle.kts` mediante los identificadores de tipo seguro disponibles en el catálogo de versiones.

   El siguiente archivo de ejemplo declara las dependencias de siete. Servicios de AWS

```
plugins {
    kotlin("jvm") version "X.Y.Z"
    application
}

group = "org.example"
version = "1.0-SNAPSHOT"

repositories {
    mavenCentral()
}

dependencies {
    implementation(platform("org.apache.logging.log4j:log4j-bom:X.Y.Z"))

    implementation(awssdk.services.s3)
    implementation(awssdk.services.dynamodb)
    implementation(awssdk.services.iam)
    implementation(awssdk.services.cloudwatch)
    implementation(awssdk.services.cognitoidentityprovider)
    implementation(awssdk.services.sns)
    implementation(awssdk.services.pinpoint)
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")

    // Test dependency.
    testImplementation(kotlin("test"))
}

tasks.test {
    useJUnitPlatform()
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(X*)
    }
```

```
}

application {
    mainClass = "org.example.AppKt"
}
```

* Versión Java, por ejemplo 17 o21.

Maven

El siguiente `pom.xml` archivo de ejemplo tiene dependencias para siete Servicios de AWS. Puede navegar hasta el *X.Y.Z* enlace para ver la última versión disponible.

```
<?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/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>setup</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <aws.sdk.kotlin.version>X.Y.Z</aws.sdk.kotlin.version>
        <kotlin.version>X.Y.Z</kotlin.version>
        <log4j.version>X.Y.Z</log4j.version>
        <junit.jupiter.version>X.Y.Z</junit.jupiter.version>
        <jvm.version>X*</jvm.version>
    </properties>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>aws.sdk.kotlin</groupId>
                <artifactId>bom</artifactId>
                <version>${aws.sdk.kotlin.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
```

```
            <dependency>
                <groupId>org.apache.logging.log4j</groupId>
                <artifactId>log4j-bom</artifactId>
                <version>${log4j.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>

        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>s3-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>dynamodb-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>iam-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>cloudwatch-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>cognitoidentityprovider-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>sns-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>pinpoint-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-slf4j2-impl</artifactId>
        </dependency>
```

```
        <!-- Test dependencies -->
        <dependency>
            <groupId>org.jetbrains.kotlin</groupId>
            <artifactId>kotlin-test-junit</artifactId>
            <version>${kotlin.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>${junit.jupiter.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <sourceDirectory>src/main/kotlin</sourceDirectory>
        <testSourceDirectory>src/test/kotlin</testSourceDirectory>

        <plugins>
            <plugin>
                <groupId>org.jetbrains.kotlin</groupId>
                <artifactId>kotlin-maven-plugin</artifactId>
                <version>${kotlin.version}</version>
                <executions>
                    <execution>
                        <id>compile</id>
                        <phase>compile</phase>
                        <goals>
                            <goal>compile</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>test-compile</id>
                        <phase>test-compile</phase>
                        <goals>
                            <goal>test-compile</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <jvmTarget>${jvm.version}</jvmTarget>
                </configuration>
```

```
            </plugin>
        </plugins>
    </build>
</project>
```

* Versión Java, por ejemplo 17 o21.

Codifica tu proyecto de Kotlin con la opción SDK para Kotlin

Ahora empieza la diversión. A medida que desarrolle su solicitud, puede consultar la AWS SDK para Kotlin APIReferencia para obtener información completa sobre las API operaciones. Usa los siguientes enlaces para obtener API información general sobre Kotlin:

- Referencia de biblioteca API estándar
- Descripción general de las corrutinas
- Corrutinas API

Inicie sesión con el AWS CLI

Siempre que ejecute un programa que acceda Servicios de AWS, necesitará una sesión activa en el portal de AWS acceso. Puede hacerlo ejecutando el comando de la .

```
aws sso login
```

Como tiene una configuración de perfil predeterminada, no necesita llamar al comando con una opción `--profile`. Si la configuración de inicio de sesión único de IAM Identity Center usa 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 de IAM Identity Center y el conjunto de permisos configurados en el `config` archivo compartido.

Configure el AWS SDK para Kotlin

En esta sección se explica cómo configurar un cliente de servicio mediante AWS SDK para Kotlin. Para obtener más información, consulte la Guía de referencia de herramientas SDK y herramientas, que incluye una descripción general de la configuración que se aplica a todos AWS SDKs.

Contenido

Crear un cliente de servicio

Para realizar una solicitud a un Servicio de AWS, primero debes crear una instancia de un cliente para ese servicio.

Puede configurar los ajustes comunes para los clientes del servicio, como el HTTP cliente que van a utilizar, el nivel de registro y la configuración de reintentos. Además, cada cliente de servicio requiere un proveedor de credenciales Región de AWS y un proveedor de credenciales. SDKUtiliza estos valores para enviar solicitudes a la región correcta y firmarlas con las credenciales correctas.

Puede especificar estos valores mediante programación en el código o hacer que se carguen automáticamente desde el entorno.

Configure un cliente en código

Para configurar un cliente de servicio con valores específicos, puede especificarlos en una función lambda que se pase al método de fábrica del cliente de servicio, como se muestra en el siguiente fragmento.

```
val dynamoDbClient = DynamoDbClient {
    region = "us-east-1"
```

```
    credentialsProvider = ProfileCredentialsProvider(profileName = "myprofile")
}
```

Todos los valores que no especifique en el bloque de configuración se establecen como valores predeterminados. Por ejemplo, si no especifica un proveedor de credenciales como lo hacía el código anterior, el proveedor de credenciales utilizará de forma predeterminada la cadena de proveedores de credenciales predeterminada.

> **Warning**
>
> Algunas propiedades, como, por ejemplo, `region` no tienen un valor predeterminado. Debe especificarlas de forma explícita en el bloque de configuración cuando utilice la configuración programática. Si no SDK puede resolver la propiedad, es posible que API las solicitudes no se realicen correctamente.

Configure un cliente desde el entorno

Al crear un cliente de servicio, SDK puede inspeccionar las ubicaciones del entorno de ejecución actual para determinar algunas propiedades de configuración. Estas ubicaciones incluyen archivos de configuración y credenciales compartidos, variables de entorno y propiedades JVM del sistema. Las propiedades disponibles para ser resueltas incluyen AWS la región, la estrategia de reintento, el modo de registro y otras. Para obtener más información sobre todos los ajustes que se SDK pueden resolver desde el entorno de ejecución, consulte la Guía de referencia de configuración de herramientas AWS SDKs y herramientas.

Para crear un cliente con una configuración basada en el entorno, utilice el método `suspend fun fromEnvironment()` estático de la interfaz del cliente del servicio:

```
val dynamoDbClient = DynamoDbClient.fromEnvironment()
```

Crear un cliente de esta manera resulta útil cuando se ejecuta en Amazon EC2 o en cualquier otro contexto en el que la configuración de un cliente de servicio esté disponible en el entorno. AWS Lambda Esto desvincula el código del entorno en el que se ejecuta y facilita la implementación de la aplicación en varias regiones sin cambiar el código.

Además, puedes anular propiedades específicas pasando un bloque lambda a. `fromEnvironment` El siguiente ejemplo carga algunas propiedades de configuración del entorno (por ejemplo, la

región), pero anula específicamente el proveedor de credenciales para usar las credenciales de un perfil.

```
val dynamoDbClient = DynamoDbClient.fromEnvironment {
    credentialsProvider = ProfileCredentialsProvider(profileName = "myprofile")
}
```

SDKUtiliza valores predeterminados para cualquier propiedad de configuración que no se pueda determinar a partir de la configuración programática o del entorno. Por ejemplo, si no especificas un proveedor de credenciales en el código o en una configuración de entorno, el proveedor de credenciales utilizará de forma predeterminada la cadena de proveedores de credenciales predeterminada.

> **Warning**
>
> Algunas propiedades, como la región, no tienen un valor predeterminado. Debe especificarlas en una configuración de entorno o de forma explícita en el bloque de configuración. Si no SDK pueden resolver la propiedad, es posible que API las solicitudes no se realicen correctamente.

> **Note**
>
> Si bien las propiedades relacionadas con las credenciales (como las claves de acceso temporales y la SSO configuración) se encuentran en el entorno de ejecución, el cliente no obtiene los valores en el momento de la creación. En cambio, la capa del proveedor de credenciales accede a los valores en cada solicitud.

Cierre el cliente

Cuando ya no necesite el cliente de servicio, ciérrelo para liberar los recursos que esté utilizando:

```
val dynamoDbClient = DynamoDbClient.fromEnvironment()
// Invoke several DynamoDB operations.
dynamoDbClient.close()
```

Como los clientes de servicio amplían la `Closeable`interfaz, puedes usar la `use`extensión para cerrar el cliente automáticamente al final de un bloque, como se muestra en el siguiente fragmento.

```
DynamoDbClient.fromEnvironment().use { dynamoDbClient ->
    // Invoke several DynamoDB operations.
}
```

En el ejemplo anterior, el bloque lambda recibe una referencia al cliente que se acaba de crear. Puede invocar operaciones en esta referencia de cliente y, cuando se complete el bloqueo (incluso mediante una excepción), el cliente estará cerrado.

Región de AWS selección

Con Regiones de AWS, puede acceder a los Servicios de AWS que operan en un área geográfica específica. Esto puede ser útil para evitar redundancias y para que sus datos y aplicaciones se ejecuten cerca del lugar desde donde usted y sus usuarios obtendrán acceso a ellos.

Cadena de proveedores por región predeterminada

Al cargar la configuración de un cliente de servicio desde el entorno, se utiliza el siguiente proceso de búsqueda:

1. Cualquier región explícita establecida en el generador.
2. La propiedad `aws.region` JVM del sistema está marcada. Si está establecida, esa región se utiliza en la configuración del cliente.
3. Se comprueba la variable de entorno `AWS_REGION`. Si está establecida, esa región se usa en la configuración del cliente.
   a. Nota: Esta variable de entorno la establece el contenedor Lambda.
4. SDKComprueba el archivo de configuración AWS compartido. Si la `region` propiedad está establecida para el perfil activo, la SDK utiliza.
   a. La variable de entorno `AWS_CONFIG_FILE` se puede utilizar para personalizar la ubicación del archivo de configuración compartida.
   b. La propiedad `aws.profile` JVM del sistema o la variable de `AWS_PROFILE` entorno se pueden utilizar para personalizar el perfil que se SDK carga.
5. Los SDK intentos de utilizar el Amazon EC2 Instance Metadata Service para determinar la región de la EC2 instancia que se está ejecutando actualmente.
6. Si la región sigue sin resolverse en este momento, se produce un error en la creación del cliente, salvo una excepción.

Proveedores de credenciales

Para realizar solicitudes a Amazon Web Services mediante AWS SDK para Kotlin, SDK utiliza credenciales firmadas criptográficamente emitidas por. AWS En tiempo de ejecución, SDK recupera los valores de configuración de las credenciales comprobando varias ubicaciones.

Si la configuración recuperada incluye la configuración de acceso de inicio de sesión único al IAM Identity Center, SDK trabaja con el IAM Identity Center para recuperar las credenciales temporales que utiliza para realizar solicitudes. Servicios de AWS

Si la configuración recuperada incluye credenciales temporales, las SDK utiliza para realizar Servicio de AWS llamadas. Las credenciales temporales constan de claves de acceso y un token de sesión.

La cadena de proveedores de credenciales predeterminada

Si no se especifica de forma explícita en la construcción del SDK cliente, Kotlin utiliza un proveedor de credenciales que comprueba secuencialmente todos los lugares en los que se pueden introducir las credenciales.

Para usar la cadena predeterminada para proporcionar las credenciales a tu aplicación, crea un cliente de servicio sin especificar explícitamente un proveedor de credenciales.

```
val ddb = DynamoDbClient {
    region = "us-east-2"
}
```

Para obtener más información, consulte las distintas formas de crear y configurar un cliente.

Orden de recuperación de credenciales

La cadena de proveedores de credenciales predeterminada busca las credenciales mediante la siguiente secuencia predefinida:

1. Variables de entorno

   Los SDK intentos de cargar las credenciales de las `AWS_ACCESS_KEY_ID` variables `AWS_SECRET_ACCESS_KEY` de `AWS_SESSION_TOKEN` entorno and y.

2. Archivos **`credentials`** y **`config`** compartidos

   Los SDK intentos de cargar las credenciales del `[default]` perfil en los `config` archivos `credentials` AND compartidos.

En este paso de secuencia, Kotlin usa el token de inicio de sesión único que se configuró al ejecutar un comando. SDK AWS CLI `aws sso login` SDKUtiliza las credenciales temporales que el Centro de IAM Identidad intercambió por un token válido. A SDK continuación, utiliza las credenciales temporales cuando llama Servicios de AWS. La Guía de referencia de AWS SDKs and Tools contiene información detallada sobre este proceso.

Puede utilizar la propiedad `aws.profile` JVM del sistema o la variable de `AWS_PROFILE` entorno para elegir el perfil que SDK desea cargar.

> **Note**
>
> Los `config` archivos `credentials` y los comparten varias AWS SDKs herramientas. Para obtener más información, consulte la. aws/credentials and .aws/configarchivos de la Guía de referencia de herramientas AWS SDKs y herramientas.

3. AWS STS identidad web (incluida Amazon Elastic Kubernetes Service (Amazon)) EKS

   Los SDK intentos de resolver las propiedades JVM del sistema y las variables de entorno para asumir un rol mediante una identidad web.

4. Credenciales de ECS contenedor de Amazon (IAMfunciones para la tarea)

   Los SDK intentos de resolver `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` o las variables de `AWS_CONTAINER_CREDENTIALS_FULL_URI` entorno de las que obtener las credenciales.

5. Amazon EC2 Instance Metadata Service (IAMfunción asociada a una instancia)

   Los SDK intentos de obtener las credenciales del servicio de metadatos de la instancia.

> **Note**
>
> El SDK único soporte IMDSv2.

6. Si las credenciales siguen sin resolverse en este momento, se produce un error en la creación del cliente, con una excepción.

Proveedor de credenciales explícitas

En lugar de usar la cadena de proveedores predeterminada, puede especificar un proveedor de credenciales específico o una cadena personalizada (`CredentialsProviderChain`) que SDK

deba usar. Por ejemplo, si establece las credenciales predeterminadas mediante variables de entorno, proporcione una `EnvironmentCredentialsProvider` al generador de clientes, tal y como se muestra en el siguiente fragmento de código.

```
val ddb = DynamoDbClient {
    region = "us-east-1"
    credentialsProvider = EnvironmentCredentialsProvider()
}
```

> **Note**
>
> La cadena predeterminada almacena en caché las credenciales, pero los proveedores independientes no. Puede agrupar cualquier proveedor de credenciales con la `CachedCredentialsProvider` clase para evitar tener que buscar credenciales innecesariamente en cada llamada. API El proveedor almacenado en caché solo obtiene las credenciales nuevas cuando las actuales caducan.

> **Note**
>
> Puede implementar su propio proveedor de credenciales o cadena de proveedores mediante la implementación de la `CredentialsProvider` interfaz.

Configure los puntos finales del cliente

Cuando AWS SDK para Kotlin llama a una Servicio de AWS, uno de sus primeros pasos es determinar dónde enrutar la solicitud. Este proceso se conoce como resolución de puntos finales.

Puede configurar la resolución de punto final para SDK cuando cree un cliente de servicio. La configuración predeterminada para la resolución de puntos finales suele ser adecuada, pero hay varios motivos que pueden llevarle a modificar la configuración predeterminada. A continuación se muestran dos ejemplos de motivos:

- Realice solicitudes a una versión preliminar de un servicio o a una implementación local de un servicio.
- Acceso a funciones de servicio específicas que aún no están modeladas en el. SDK

> **Warning**
>
> La resolución de terminales es un SDK tema avanzado. Si cambias la configuración predeterminada, corres el riesgo de infringir el código. La configuración predeterminada debería aplicarse a la mayoría de los usuarios en entornos de producción.

Configuración personalizada

Puede personalizar la resolución de punto final de un cliente de servicio con dos propiedades que están disponibles al crear el cliente:

1. `endpointUrl: Url`
2. `endpointProvider: EndpointProvider`

Establezca **`endpointUrl`**

Puede establecer un valor `endpointUrl` para indicar un nombre de host «base» para el servicio. Sin embargo, este valor no es definitivo, ya que se pasa como parámetro a la `EndpointProvider` instancia del cliente. Luego, la `EndpointProvider` implementación puede inspeccionar y, posiblemente, modificar ese valor para determinar el punto final.

Por ejemplo, si especificas un `endpointUrl` valor para un cliente de Amazon Simple Storage Service (Amazon S3) y realizas `GetObject` una operación, la implementación del proveedor de puntos de conexión predeterminado inserta el nombre del bucket en el valor del nombre de host.

En la práctica, los usuarios establecen un `endpointUrl` valor para que apunte a una instancia de desarrollo o vista previa de un servicio.

Establezca **`endpointProvider`**

La `EndpointProvider` implementación de un cliente de servicio determina la resolución final del punto final. La `EndpointProvider` interfaz que se muestra en el siguiente bloque de código expone el `resolveEndpoint` método.

```
public fun interface EndpointProvider<T> {
    public suspend fun resolveEndpoint(params: T): Endpoint
}
```

Un cliente de servicio llama al `resolveEndpoint` método para cada solicitud. El cliente del servicio utiliza el `Endpoint` valor devuelto por el proveedor sin más cambios.

Propiedades de **`EndpointProvider`**

El `resolveEndpoint` método acepta un `EndpointParameters` objeto específico del servicio que contiene las propiedades utilizadas en la resolución de puntos finales.

Cada servicio incluye las siguientes propiedades base.

Nombre	Tipo	Descripción
`region`	Cadena	La AWS región del cliente
`endpoint`	Cadena	Una representación en cadena del conjunto de valores de `endpointUrl`
`useFips`	Booleano	Si FIPS los puntos finales están habilitados en la configuración del cliente
`useDualStack`	Booleano	Si los puntos finales de doble pila están habilitados en la configuración del cliente

Los servicios pueden especificar las propiedades adicionales necesarias para la resolución. Por ejemplo, Amazon S3 `S3EndpointParameters`incluye el nombre del bucket y también varios ajustes de funciones específicos de Amazon S3. Por ejemplo, la `forcePathStyle` propiedad determina si se puede utilizar el direccionamiento del host virtual.

Si implementas tu propio proveedor, no deberías necesitar crear tu propia instancia de`EndpointParameters`. SDKProporciona las propiedades de cada solicitud y las pasa a su implementación de`resolveEndpoint`.

`endpointUrl` o **`endpointProvider`**

Es importante entender que las dos afirmaciones siguientes NOT producen clientes con un comportamiento de resolución de punto final equivalente:

```
// Use endpointUrl.
S3Client.fromEnvironment {
    endpointUrl = Url.parse("https://endpoint.example")
}

// Use endpointProvider.
S3Client.fromEnvironment {
    endpointProvider = object : S3EndpointProvider {
        override suspend fun resolveEndpoint(params: S3EndpointParameters): Endpoint =
 Endpoint("https://endpoint.example")
    }
}
```

La sentencia que establece la `endpointUrl` propiedad especifica una base URL que se pasa al proveedor (predeterminado), que se puede modificar como parte de la resolución del punto final.

La declaración que establece el `endpointProvider` especifica el final que URL se `S3Client` utiliza.

Aunque puede establecer ambas propiedades, en la mayoría de los casos en las que es necesario personalizarlas, debe proporcionar una de ellas. Como SDK usuario general, la mayoría de las veces proporciona un `endpointUrl` valor.

Nota sobre Amazon S3

Amazon S3 es un servicio complejo con muchas de sus características modeladas a través de personalizaciones de puntos de enlace personalizadas, como el alojamiento virtual por cubos. El alojamiento virtual es una función de Amazon S3 en la que el nombre del bucket se inserta en el nombre del host.

Por este motivo, le recomendamos que no sustituya la `EndpointProvider` implementación en un cliente de servicio Amazon S3. Si necesita ampliar su comportamiento de resolución, por ejemplo enviando solicitudes a una pila de desarrollo local teniendo en cuenta otros aspectos relacionados con los puntos finales, le recomendamos empaquetar la implementación predeterminada. El siguiente `endpointProvider` ejemplo muestra un ejemplo de implementación de este enfoque.

Ejemplos

endpointUrlEjemplo de

El siguiente fragmento de código muestra cómo se puede anular el punto final de servicio general para un cliente de Amazon S3.

```
val client = S3Client.fromEnvironment {
    endpointUrl = Url.parse("https://custom-s3-endpoint.local")
    // EndpointProvider is left as the default.
}
```

endpointProviderEjemplo de

En el siguiente fragmento de código se muestra cómo proporcionar un proveedor de puntos de conexión personalizado que incluya la implementación predeterminada de Amazon S3.

```
import aws.sdk.kotlin.services.s3.endpoints.DefaultS3EndpointProvider
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointParameters
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointProvider
import aws.smithy.kotlin.runtime.client.endpoints.Endpoint

public class CustomS3EndpointProvider : S3EndpointProvider {
    override suspend fun resolveEndpoint(params: S3EndpointParameters) =
        if (/* Input params indicate we must route another endpoint for whatever
 reason. */) {
            Endpoint(/* ... */)
        } else {
            // Fall back to the default resolution.
            DefaultS3EndpointProvider().resolveEndpoint(params)
        }
}
```

endpointUrl y **endpointProvider**

El siguiente programa de ejemplo demuestra la interacción entre la `endpointUrl` configuración y. `endpointProvider` Se trata de un caso de uso avanzado.

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.endpoints.DefaultS3EndpointProvider
```

```
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointParameters
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointProvider
import aws.smithy.kotlin.runtime.client.endpoints.Endpoint

fun main() = runBlocking {
    S3Client.fromEnvironment {
        endpointUrl = Url.parse("https://example.endpoint")
        endpointProvider = CustomS3EndpointProvider()
    }.use { s3 ->
        // ...
    }
}

class CustomS3EndpointProvider : S3EndpointProvider {
    override suspend fun resolveEndpoint(params: S3EndpointParameters) {
        // The resolved string value of the endpointUrl set in the client above is
 available here.
        println(params.endpoint)
        // ...
    }
}
```

HTTP

En esta sección se describe la configuración de HTTP los ajustes relacionados en AWS SDK para Kotlin.

Temas

- HTTPconfiguración del cliente
- Usa un HTTP proxy
- HTTPinterceptores
- Imponga una versión mínima TLS

HTTPconfiguración del cliente

De forma predeterminada, AWS SDK para Kotlin utiliza un HTTP cliente basado en OkHttp. Puede anular el HTTP cliente y su configuración proporcionando un cliente configurado de forma explícita.

> **Note**
>
> De forma predeterminada, cada cliente de servicio usa su propia copia de un HTTP cliente. Si usa varios servicios en la aplicación, es posible que desee crear un único HTTP cliente y compartirlo entre todos los clientes del servicio.

Configuración básica

Al configurar un cliente de servicio, puede configurar el tipo de motor predeterminado. SDKAdministra el motor del HTTP cliente resultante y lo cierra automáticamente cuando ya no es necesario.

El siguiente ejemplo muestra la configuración de un HTTP cliente durante la inicialización de un cliente de DynamoDB.

Importaciones

```
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import kotlin.time.Duration.Companion.seconds
```

Código

```
DynamoDbClient {
    region = "us-east-2"
    httpClient {
        maxConcurrency = 64u
        connectTimeout = 10.seconds
    }
}.use { ddb ->

    // Perform some actions with Amazon DynamoDB.
}
```

Especifique un tipo de motor HTTP

Para casos de uso más avanzados, puede pasar un parámetro adicional `httpClient` que especifique el tipo de motor. De esta forma, puede establecer parámetros de configuración exclusivos para ese tipo de motor.

En el siguiente ejemplo se especifica lo `OkHttpEngine`que puede utilizar para configurar la `maxConcurrencyPerHost`propiedad.

Importaciones

```
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import aws.smithy.kotlin.runtime.http.engine.okhttp.OkHttpEngine
```

Código

```
DynamoDbClient {
    region = "us-east-2"
    httpClient(OkHttpEngine) {  // The first parameter specifies the HTTP engine type.
        // The following parameter is generic HTTP configuration available in any
 engine type.
        maxConcurrency = 64u

            // The following parameter is OkHttp-specific configuration.
        maxConcurrencyPerHost = 32u
    }
}.use { ddb ->

    // Perform some actions with Amazon DynamoDB.
}
```

Los valores posibles para el tipo de motor son `OkHttpEngine` `OkHttp4Engine`, y `CrtHttpEngine`.

Para utilizar los parámetros de configuración específicos de un HTTP motor, debe añadir el motor como una dependencia en tiempo de compilación. Para ello`OkHttpEngine`, agrega la siguiente dependencia mediante Gradle.

(Puedes navegar hasta el *`X.Y.Z`* enlace para ver la última versión disponible).

```
implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
implementation("aws.smithy.kotlin:http-client-engine-okhttp")
```

Para el`CrtHttpEngine`, añada la siguiente dependencia.

```
implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
```

```
implementation("aws.smithy.kotlin:http-client-engine-crt")
```

Utilizar **`OkHttp4Engine`**

Use la opción predeterminada `OkHttp4Engine` si no puede usar la opción predeterminada`OkHttpEngine`. El GitHub repositorio smithy-kotlin tiene información sobre cómo configurar y usar el. `OkHttp4Engine`

Usa un cliente explícito HTTP

Cuando utilizas un HTTP cliente explícito, eres responsable de su duración, incluido el cierre cuando ya no lo necesites. Un HTTP cliente debe vivir al menos tanto como cualquier cliente del servicio que lo utilice.

El siguiente ejemplo de código muestra el código que mantiene HTTP al cliente activo mientras `DynamoDbClient` está activo. La `use`función se asegura de que el HTTP cliente se cierre correctamente.

Importaciones

```
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import aws.smithy.kotlin.runtime.http.engine.okhttp.OkHttpEngine
import kotlin.time.Duration.Companion.seconds
```

Código

```
OkHttpEngine {
    maxConcurrency = 64u
    connectTimeout = 10.seconds
}.use { okHttpClient ->

     DynamoDbClient {
        region = "us-east-2"
        httpClient = okHttpClient
    }.use { ddb ->
        {
            // Perform some actions with Amazon DynamoDB.
        }
    }
}
```

Usa un HTTP proxy

Para acceder AWS a través de servidores proxy mediante el AWS SDK para Kotlin, puede configurar las propiedades JVM del sistema o las variables de entorno. Si se proporcionan ambas, las propiedades JVM del sistema tienen prioridad.

Utilice las propiedades JVM del sistema

SDKBusca las propiedades del JVM sistema `https.proxyHosthttps.proxyPort`, y`http.nonProxyHosts`. Para obtener más información sobre estas propiedades comunes JVM del sistema, consulte Redes y proxies en la documentación de Java.

```
java -Dhttps.proxyHost=10.15.20.25 -Dhttps.proxyPort=1234 -
Dhttp.nonProxyHosts=localhost|api.example.com MyApplication
```

Utilización de variables de entorno

SDKBusca las variables de `no_proxy` entorno `https_proxyhttp_proxy`, y las versiones en mayúscula de cada una de ellas.

```
export http_proxy=http://10.15.20.25:1234
export https_proxy=http://10.15.20.25:5678
export no_proxy=localhost,api.example.com
```

Usa un proxy en las instancias EC2

Si configuras un proxy en una EC2 instancia lanzada con un IAM rol asociado, asegúrate de eximir la dirección que se usa para acceder a los metadatos de la instancia. Para ello, establece la propiedad del `http.nonProxyHosts` JVM sistema o la variable de `no_proxy` entorno en la dirección IP del servicio de metadatos de la instancia, que es`169.254.169.254`. Esta dirección no varía.

```
export no_proxy=169.254.169.254
```

HTTPinterceptores

Puede utilizar interceptores para facilitar la ejecución de API solicitudes y respuestas. Los interceptores son mecanismos abiertos en los que SDK llaman al código que se escribe para

introducir un comportamiento en el ciclo de vida de la solicitud/respuesta. De esta forma, puedes modificar una solicitud en curso, depurar el procesamiento de una solicitud, ver las excepciones y mucho más.

El siguiente ejemplo muestra un interceptor simple que añade un encabezado adicional a todas las solicitudes salientes antes de entrar en el bucle de reintentos.

```
class AddHeader(
    private val key: String,
    private val value: String
) : HttpInterceptor {
    override suspend fun modifyBeforeRetryLoop(context:
 ProtocolRequestInterceptorContext<Any, HttpRequest>): HttpRequest {
        val httpReqBuilder = context.protocolRequest.toBuilder()
        httpReqBuilder.headers[key] = value
        return httpReqBuilder.build()
    }
}
```

Para obtener más información y conocer los ganchos de intercepción disponibles, consulte la interfaz Interceptor.

Registro del interceptor

Los interceptores se registran cuando se crea un cliente de servicio o cuando se anula la configuración de un conjunto específico de operaciones.

Interceptor para todas las operaciones del cliente de servicio

El siguiente código añade una `AddHeader` instancia a la propiedad de interceptores del generador. Esta adición añade el `x-foo-version` encabezado a todas las operaciones antes de entrar en el bucle de reintento.

```
val s3 = S3Client.fromEnvironment {
    interceptors += AddHeader("x-foo-version", "1.0")
}

// All service operations invoked using 's3' will have the header appended.
s3.listBuckets { ... }
s3.listObjectsV2 { ... }
```

Interceptor solo para operaciones específicas

Al utilizar la `withConfig` extensión, puede anular la configuración del cliente de servicio para una o más operaciones de cualquier cliente de servicio. Con esta capacidad, puede registrar interceptores adicionales para un subconjunto de operaciones.

El siguiente ejemplo anula la configuración de la `s3` instancia para las operaciones dentro de la extensión. `use` Las operaciones ejecutadas `s3Scoped` contienen tanto los encabezados como `x-foo-version` los `x-bar-version` encabezados.

```
// 's3' instance created in the previous code snippet.
s3.withConfig {
    interceptors += AddHeader("x-bar-version", "3.7")
}.use { s3Scoped ->
    // All service operations invoked using 's3Scoped' trigger interceptors
    // that were registered when the client was created and any added in the
    // withConfig { ... } extension.
}
```

Imponga una versión mínima TLS

Con el AWS SDK para Kotlin, puede configurar la TLS versión mínima cuando se conecte a los puntos finales del servicio. SDKOfrece diferentes opciones de configuración. En orden de mayor a menor prioridad, las opciones son:

- Configure el motor de forma explícita HTTP
- Establezca la propiedad `sdk.minTls` JVM del sistema
- Defina la variable de `SDK_MIN_TLS` entorno

Configure el HTTP motor

Al especificar un HTTP motor no predeterminado para un cliente de servicio, puede configurar el `tlsContext.minVersion` campo.

El siguiente ejemplo configura el HTTP motor y cualquier cliente de servicio que lo utilice para que utilicen la TLS v1.2 como mínimo.

```
DynamoDbClient {
```

```
    region = "us-east-2"
    httpClient {
        tlsContext {
            minVersion = TlsVersion.TLS_1_2
        }
    }
}.use { ddb ->

    // Perform some actions with Amazon DynamoDB.
}
```

Establezca la propiedad del sistema **`sdk.minTls`** JVM

Puede establecer la propiedad `sdk.minTls` JVM del sistema. Al iniciar una aplicación con las propiedades del sistema definidas, todos los HTTP motores construidos con ella AWS SDK para Kotlin utilizan la TLS versión mínima especificada de forma predeterminada. Sin embargo, puede anular esto de forma explícita en la configuración del HTTP motor. Los valores permitidos son:

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

Defina la variable de entorno **`SDK_MIN_TLS`**

Puede configurar la variable de `SDK_MIN_TLS` entorno. Al lanzar una aplicación con la variable de entorno configurada, todos los HTTP motores construidos con ella AWS SDK para Kotlin utilizan la TLS versión mínima especificada, a menos que se sustituya por otra opción.

Los valores permitidos son:

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

Reintentos

Las llamadas devuelven Servicios de AWS ocasionalmente excepciones inesperadas. Algunos tipos de errores, como los errores transitorios o de limitación, pueden producirse correctamente si se vuelve a intentar realizar la llamada.

En esta página se describe cómo configurar los reintentos automáticos con. AWS SDK para Kotlin

Configuración de reintentos predeterminada

De forma predeterminada, cada cliente de servicio se configura automáticamente con una estrategia de reintentos estándar. La configuración predeterminada intenta realizar una llamada que falla hasta tres veces (el intento inicial más dos reintentos). El retardo intermedio entre cada llamada se configura con un retardo exponencial y una fluctuación aleatoria para evitar tormentas de reintentos. Esta configuración funciona en la mayoría de los casos de uso, pero puede no ser adecuada en algunas circunstancias, como en los sistemas de alto rendimiento.

Los SDK intentos se reintentan solo en caso de errores que se puedan 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. Los parámetros faltantes o no válidos, los errores de autenticación o seguridad y las excepciones de mala configuración no se consideran reintentables.

Puede personalizar la estrategia de reintentos estándar estableciendo el número máximo de intentos, las demoras y los retrasos, así como la configuración del conjunto de fichas.

Número máximo de intentos

Puede personalizar el número máximo de intentos predeterminado (3) en el `retryStrategyDSLbloque` durante la construcción del cliente.

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        maxAttempts = 5
    }
}
```

Con el cliente del servicio DynamoDB mostrado en el fragmento anterior, API intenta realizar llamadas que fallan hasta cinco veces (SDKel intento inicial más cuatro reintentos).

Para deshabilitar completamente los reintentos automáticos, establezca el número máximo de intentos en uno, como se muestra en el siguiente fragmento.

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        maxAttempts = 1  // The SDK makes no retries.
    }
}
```

Retrasos y retrasos

Si es necesario volver a intentarlo, la estrategia de reintento predeterminada espera antes de realizar el siguiente intento. El retraso del primer reintento es pequeño, pero aumenta exponencialmente en los reintentos posteriores. La cantidad máxima de retraso está limitada para que no aumente demasiado.

Por último, se aplica una fluctuación aleatoria a los retrasos entre todos los intentos. La fluctuación ayuda a mitigar el efecto de las grandes flotas, que pueden provocar tormentas que vuelvan a intentarlo. (Consulte esta entrada del blog de AWS arquitectura para obtener un análisis más profundo sobre el retroceso y la fluctuación exponenciales).

Los parámetros de retardo se pueden configurar en el bloque. `delayProvider` DSL

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        delayProvider {
            initialDelay = 100.milliseconds
            maxBackoff = 5.seconds
        }
    }
}
```

Con la configuración que se muestra en el fragmento anterior, el cliente retrasa el primer intento hasta 100 milisegundos. El tiempo máximo entre un reintento es de 5 segundos.

Están disponibles los siguientes parámetros para los retrasos y el retardo del ajuste.

Parámetro	Valor predeterminado	Descripción
`initialDelay`	10 milisegundos	La cantidad máxima de retraso para el primer

Parámetro	Valor predeterminado	Descripción
		reintento. Cuando se aplica una fluctuación de fase, la cantidad real de retraso puede ser menor.
`jitter`	1.0 (fluctuación total)	La amplitud máxima con la que se puede reducir aleatoriamente el retraso calculado. El valor predeterminado de 1,0 significa que el retraso calculado se puede reducir hasta un 100% (por ejemplo, hasta 0). Un valor de 0,5 significa que el retraso calculado se puede reducir hasta la mitad. Por lo tanto, un retraso máximo de 10 ms podría reducirse a entre 5 ms y 10 ms. Un valor de 0.0 significa que no se aplica ninguna fluctuación. **Important** # La configuración de fluctuación es una función avanzada. Por lo general, no se recomienda personalizar este comportamiento.

Parámetro	Valor predeterminado	Descripción
`maxBackoff`	20 segundos	La cantidad máxima de demora que se puede aplicar a cualquier intento. Si se establece este valor, se limita el crecimiento exponenci al que se produce entre los intentos posteriores y se evita que el máximo calculado sea demasiado grande. Este parámetro limita el retraso calculado antes de que se aplique la fluctuación de fase. Si se aplica, la fluctuación podría reducir aún más el retraso.

Parámetro	Valor predeterminado	Descripción
`scaleFactor`	1.5	La base exponencial a partir de la cual se increment arán los retardos máximos subsiguientes. Por ejemplo, si uno es `initialDelay` de 10 ms y uno `scaleFact or` de 1,5, se calcularían los siguientes retrasos máximos: • Reintento 1:10 ms × 1,5° = 10 ms • Reintento 2:10 ms × 1,5¹ = 15 ms • Reintento 3:10 ms × 1,5² = 22,5 ms • Reintento 4:10 ms × 1,5³ = 33,75 ms Cuando se aplica una fluctuaci ón, la cantidad real de cada retraso puede ser menor.

Vuelva a intentar el depósito de fichas

Puede modificar aún más el comportamiento de la estrategia de reintento estándar ajustando la configuración predeterminada del depósito de fichas. El conjunto de fichas de reintentos ayuda a reducir los reintentos que tienen menos probabilidades de éxito o que pueden tardar más tiempo en resolverse, como los errores de tiempo de espera o de aceleración.

> **Important**
>
> La configuración del depósito de fichas es una función avanzada. Por lo general, no se recomienda personalizar este comportamiento.

Cada reintento (incluido opcionalmente el intento inicial) reduce parte de la capacidad del depósito de fichas. La cantidad disminuida depende del tipo de intento. Por ejemplo, volver a intentar los errores transitorios puede resultar económico, pero volver a intentarlo cuando se agota el tiempo de espera o se reduce el tiempo de espera puede resultar más caro.

Si el intento se realiza correctamente, se devuelve la capacidad al depósito. El depósito no puede incrementarse más allá de su capacidad máxima ni reducirse por debajo de cero.

Según el valor de la `useCircuitBreakerMode` configuración, los intentos de reducir la capacidad por debajo de cero producen uno de los siguientes resultados:

- Si la configuración esTRUE, se produce una excepción: por ejemplo, si se han realizado demasiados reintentos y es poco probable que se realicen más reintentos.
- Si la configuración esFALSE, se produce un retraso (por ejemplo, se retrasa hasta que el depósito vuelva a tener suficiente capacidad).

Los parámetros del depósito de fichas se pueden configurar en el `tokenBucketDSLbloque`:

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        tokenBucket {
            maxCapacity = 100
            refillUnitsPerSecond = 2
        }
    }
}
```

Los siguientes parámetros están disponibles para ajustar el grupo de fichas de reintentos:

Parámetro	Valor predeterminado	Descripción
`initialTryCost`	0	La cantidad que se va a reducir con respecto a la cubeta en los intentos iniciales . El valor predeterminado de 0 significa que no se reducirá la capacidad y, por lo tanto,

Parámetro	Valor predeterminado	Descripción
		los intentos iniciales no se detendrán ni retrasarán.
`initialTrySuccessIncrement`	1	La cantidad necesaria para incrementar la capacidad cuando el intento inicial se realizó correctamente.
`maxCapacity`	500	La capacidad máxima del depósito de fichas. El número de fichas disponibles no puede superar este número.
`refillUnitsPerSecond`	0	La cantidad de capacidad que se vuelve a añadir al depósito cada segundo. Un valor de 0 significa que no se vuelve a añadir capacidad automáticamente. (Por ejemplo, solo los intentos exitosos dan como resultado un aumento de la capacidad). Un valor de 0 debe `useCircuitBreakerMode` ser. TRUE
`retryCost`	5	La cantidad que se debe reducir del intervalo en caso de un intento tras un error transitorio. La misma cantidad se vuelve a acumular en el depósito si el intento tiene éxito.

Parámetro	Valor predeterminado	Descripción
`timeoutRetryCost`	10	La cantidad que se deduce de la cubeta para un intento tras un fallo en el tiempo de espera o en la aceleración. La misma cantidad se vuelve a acumular en el depósito si el intento tiene éxito.
`useCircuitBreakerMode`	TRUE	Determina el comportamiento en caso de que un intento de reducir la capacidad del depósito caiga por debajo de cero. CuandoTRUE, el depósito de fichas generará una excepción que indicará que no existe más capacidad de reintentos. FALSEEn ese caso, el depósito de fichas retrasará el intento hasta que se haya rellenado la capacidad suficiente.

Reintentos adaptativos

Como alternativa a la estrategia de reintentos estándar, la estrategia de reintentos adaptativa es un enfoque avanzado que busca la tasa de solicitudes ideal para minimizar los errores de limitación.

> **Important**
>
> Los reintentos adaptativos son un modo de reintento avanzado. Por lo general, no se recomienda utilizar esta estrategia de reintentos.

Los reintentos adaptables incluyen todas las características de los reintentos estándar. Añade un limitador de velocidad en el lado del cliente que mide la tasa de solicitudes restringidas en

comparación con las solicitudes no restringidas. También limita el tráfico para intentar mantenerse dentro de un ancho de banda seguro, lo que idealmente no provoca errores de limitación.

La velocidad se adapta en tiempo real a los cambios en las condiciones del servicio y los patrones de tráfico y, en consecuencia, podría aumentar o disminuir la velocidad del tráfico. Lo que es más grave, el limitador de velocidad podría retrasar los intentos iniciales en situaciones de alto tráfico.

La estrategia de reintento adaptativo se selecciona proporcionando un parámetro adicional al método. `retryStrategy` Los parámetros del limitador de velocidad se pueden configurar en el `rateLimiter` DSL bloque.

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy(AdaptiveRetryStrategy) {
        maxAttempts = 10
        rateLimiter {
            minFillRate = 1.0
            smoothing = 0.75
        }
    }
}
```

> **Note**
>
> 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.

Observabilidad

La observabilidad es la medida en que se puede deducir el estado actual de un sistema a partir de los datos que emite. Los datos emitidos se denominan comúnmente telemetría.

AWS SDK para Kotlin Pueden proporcionar las tres señales de telemetría más comunes: métricas, trazas y registros. Puedes configurar un cable `TelemetryProvider`para enviar datos de telemetría a un backend de observabilidad (como AWS X-RayAmazon CloudWatch) y luego actuar en consecuencia.

De forma predeterminada, solo el registro está habilitado y otras señales de telemetría están deshabilitadas en el. SDK En este tema se explica cómo habilitar y configurar la salida de telemetría.

> **Important**
>
> `TelemetryProvider`actualmente es un experimento API que debe habilitarse para su uso.

Configure un `TelemetryProvider`

Puede configurar un `TelemetryProvider` en su aplicación de forma global para todos los clientes del servicio o para clientes individuales. Los siguientes ejemplos utilizan una `getConfiguredProvider()` función hipotética para demostrar las `TelemetryProvider` API operaciones. the section called “Proveedores de telemetría”En la sección se describe la información sobre las implementaciones proporcionada por. SDK Si un proveedor no es compatible, puedes implementar tu propio soporte o abrir una solicitud de función en GitHub.

Configura el proveedor de telemetría global predeterminado

De forma predeterminada, todos los clientes del servicio intentan utilizar el proveedor de telemetría disponible en todo el mundo. De esta forma, puede configurar el proveedor una vez y todos los clientes lo usarán. Esto debe hacerse solo una vez, antes de crear una instancia de cualquier cliente de servicio.

Para usar el proveedor de telemetría global, primero actualiza las dependencias del proyecto para agregar el módulo de telemetría predeterminado, como se muestra en el siguiente fragmento de código de Gradle.

(Puedes navegar hasta el enlace para ver la *`X.Y.Z`* última versión disponible).

```
dependencies {
    implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
    implementation("aws.smithy.kotlin:telemetry-defaults")
    ...
}
```

A continuación, configure el proveedor de telemetría global antes de crear un cliente de servicio, como se muestra en el siguiente código.

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.smithy.kotlin.runtime.telemetry.GlobalTelemetryProvider
import kotlinx.coroutines.runBlocking

fun main() = runBlocking {
    val myTelemetryProvider = getConfiguredProvider()
    GlobalTelemetryProvider.set(myTelemetryProvider)

    S3Client.fromEnvironment().use { s3 ->
        …
    }
}

fun getConfiguredProvider(): TelemetryProvider {
    TODO("TODO - configure a provider")
}
```

Configure un proveedor de telemetría para un cliente de servicio específico

Puede configurar un cliente de servicio individual con un proveedor de telemetría específico (distinto del global). Esto se muestra en el siguiente ejemplo.

```
import aws.sdk.kotlin.services.s3.S3Client
import kotlinx.coroutines.runBlocking

fun main() = runBlocking {
    S3Client.fromEnvironment{
        telemetryProvider = getConfiguredProvider()
    }.use { s3 ->
        ...
    }
}

fun getConfiguredProvider(): TelemetryProvider {
    TODO("TODO - configure a provider")
}
```

Métricas

En la siguiente tabla se enumeran las métricas de telemetría que emite. SDK Configure un proveedor de telemetría para que las métricas sean observables.

¿Qué métricas se emiten?

Nombre de métrica	Unidades	Tipo	Atributos	Descripción
smithy.client.call.duration	s	Histograma	rpc.service, rpc.método	Duración total de la llamada (incluidos los reintentos)
smithy.client.call.attempts	{intento}	MonotonCounter	rpc.service, rpc.método	El número de intentos de una operación individual
smithy.client.call.errors	{error}	MonotonCounter	rpc.service, rpc.método, exception.type	El número de errores de una operación
smithy.client.call.attempt_duration	s	Histograma	rpc.service, rpc.método	El tiempo que se tarda en conectarse al servicio, enviar la solicitud y recuperar el código de HTTP estado y los encabezados (incluido el tiempo de espera en la cola para ser enviados)
smithy.client.call.resolve_endpoint_duration	s	Histograma	rpc.service, rpc.método	El tiempo que se tarda en resolver un punto final (no DNS el solucionador de puntos finales) de la solicitud
smithy.client.call.serialization_duration	s	Histograma	rpc.service, rpc.método	El tiempo que se tarda en serializar el cuerpo de un mensaje
smithy.client.call.deserialization_duration	s	Histograma	rpc.service, rpc.método	El tiempo que se tarda en deserializar el cuerpo de un mensaje
smithy.client.call.auth.signing_duration	s	Histograma	rpc.service, rpc.method, auth.scheme_id	El tiempo que se tarda en firmar una solicitud

Nombre de métrica	Unidades	Tipo	Atributos	Descripción
smithy.client.call.auth.resolve_identity_duration	s	Histograma	rpc.service, rpc.method, auth.scheme_id	El tiempo que se tarda en adquirir una identidad (como AWS credenciales o un token portador) de un proveedor de identidades
smithy.client.http.connections.acquire_duration	s	Histograma		El tiempo que tarda una solicitud en adquirir una conexión
smithy.client.http.connections.limit	{conexión}	[Asincrónico] UpDownnter		El número máximo de conexiones abiertas permitidas/configuradas para el cliente HTTP
smithy.client.http.connections.usage	{conexión}	[Asincrónico] UpDownnter	estado: inactivo | adquirido	Estado actual del pool de conexiones
smithy.client.http.connections.uptime	s	Histograma		El tiempo que ha estado abierta una conexión
smithy.client.http.requests.usage	{solicitud}	[Asincrónico] UpDownnter	estado: en cola | en vuelo	El estado actual de la simultaneidad de solicitudes del HTTP cliente
smithy.client.http.requests.queued_duration	s	Histograma		La cantidad de tiempo que una solicitud pasó en cola y esperando a que el cliente la ejecutara HTTP

Nombre de métrica	Unidades	Tipo	Atributos	Descripción
smithy.client.http.bytes_sent	Mediante	Monoton Counter	dirección del servidor	El número total de bytes enviados por el cliente HTTP
smithy.client.http.bytes_received	Mediante	Monoton Counter	dirección del servidor	El número total de bytes recibidos por el cliente HTTP

A continuación se muestran las descripciones de las columnas:

- Nombre de la métrica: el nombre de la métrica emitida.
- Unidades: la unidad de medida de la métrica. Las unidades se indican en notación que distingue UCUMmayúsculas de minúsculas («c/s»).
- Tipo: el tipo de instrumento utilizado para capturar la métrica.
- Descripción: una descripción de lo que mide la métrica.
- Atributos: el conjunto de atributos (dimensiones) emitidos con la métrica.

Registro

AWS SDK para Kotlin Configura un registrador SLF4Jcompatible como predeterminado del proveedor `LoggerProvider` de telemetría. ConSLF4J, que es una capa de abstracción, puede utilizar cualquiera de los varios sistemas de registro en tiempo de ejecución. Los sistemas de registro compatibles incluyen Java Logging APIs, Log4j 2 y Logback.

> **Warning**
>
> Le recomendamos que utilice el registro de cables únicamente con fines de depuración. (El registro de cables se analiza más adelante). Desactívelo en sus entornos de producción porque puede registrar datos confidenciales, como direcciones de correo electrónico, identificadores de seguridad, API claves, contraseñas y AWS Secrets Manager secretos. El registro de transferencias registra la solicitud o respuesta completa sin cifrar, incluso cuando se trata de una HTTPS llamada.

En el caso de solicitudes o respuestas de gran tamaño (como subir un archivo a Amazon S3), el registro detallado de las conexiones también puede afectar considerablemente al rendimiento de la aplicación.

Ejemplo de configuración de registro de Log4j 2

Si bien se puede utilizar cualquier biblioteca de registros `SLF4J` compatible, este ejemplo permite la salida de registros desde los JVM programas integrados que utilizan SDK Log4j 2:

Dependencias de Gradle

(Puedes navegar hasta el *`X.Y.Z`* enlace para ver la última versión disponible).

```
implementation("org.apache.logging.log4j:log4j-slf4j2-impl:X.Y.Z")
```

Archivo de configuración de Log4j 2

Cree un archivo con un nombre `log4j2.xml` en su `resources` directorio (por ejemplo,`<project-dir>/src/main/resources`). Añada la siguiente XML configuración al archivo:

```
<Configuration status="ERROR">
    <Appenders>
        <Console name="Out">
            <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} %-5p %c:%L %X - %encode{%m}
{CRLF}%n"/>
        </Console>
    </Appenders>
    <Loggers>
        <Root level="info">
            <AppenderRef ref="Out"/>
        </Root>
    </Loggers>
</Configuration>
```

Esta configuración incluye el %X especificador en el `pattern` atributo que permite el registro MDC (contexto de diagnóstico mapeado).

SDKAgrega los siguientes MDC elementos para cada operación.

rpc

El nombre de la persona invocadaRPC, por ejemplo`S3.GetObject`.

sdkInvocationId

Un identificador único asignado por el cliente del servicio para la operación. El ID correlaciona todos los eventos de registro relacionados con la invocación de una sola operación.

Especifique el modo de registro para los mensajes a nivel de cable

De forma predeterminada, AWS SDK para Kotlin no registra los mensajes a nivel de cable porque pueden contener datos confidenciales de solicitudes y respuestasAPI. Sin embargo, a veces se necesita este nivel de detalle para realizar tareas de depuración.

Con KotlinSDK, puedes configurar un modo de registro en el código o usar la configuración del entorno para habilitar los mensajes de depuración en los siguientes casos:

- Solicitudes HTTP
- HTTP respuestas

El modo de registro está respaldado por un campo de bits en el que cada bit es un indicador (modo) y los valores son aditivos. Puede combinar un modo de solicitud y un modo de respuesta.

Configure el modo de registro en el código

Para optar por un registro adicional, defina la `logMode` propiedad cuando cree un cliente de servicio.

El siguiente ejemplo muestra cómo habilitar el registro de las solicitudes (con el cuerpo) y la respuesta (sin el cuerpo).

```
import aws.smithy.kotlin.runtime.client.LogMode

// ...

val client = DynamoDbClient {
    // ...
    logMode = LogMode.LogRequestWithBody + LogMode.LogResponse
}
```

Un valor de modo de registro establecido durante la construcción del cliente de servicio anula cualquier valor de modo de registro establecido en el entorno.

Establezca el modo de registro desde el entorno

Para establecer un modo de registro global para todos los clientes de servicio que no estén configurados explícitamente en el código, utilice una de las siguientes opciones:

- JVMpropiedad del sistema: `sdk.logMode`
- Variable de entorno: `SDK_LOG_MODE`

Están disponibles los siguientes valores que no distinguen mayúsculas de minúsculas:

- `LogRequest`
- `LogRequestWithBody`
- `LogResponse`
- `LogResponseWithBody`

Para crear un modo de registro combinado utilizando la configuración del entorno, separe los valores con un símbolo de barra vertical (`|`).

Por ejemplo, en los ejemplos siguientes se establece el mismo modo de registro que en el ejemplo anterior.

```
# Environment variable.
export SDK_LOG_MODE=LogRequestWithBody|LogResponse
```

```
# JVM system property.
java -Dsdk.logMode=LogRequestWithBody|LogResponse ...
```

> **Note**
>
> También debe configurar un SLF4J registrador compatible y establecer el nivel de registro en para habilitar el registro DEBUG a nivel de cable.

Proveedores de telemetría

SDKActualmente es compatible con OpenTelemetry(OTel) como proveedor. SDKEs posible que ofrezcan proveedores de telemetría adicionales en el futuro.

Temas

- Configure el proveedor de OpenTelemetry telemetría basado

Configure el proveedor de OpenTelemetry telemetría basado

El SDK para Kotlin proporciona una implementación de la `TelemetryProvider` interfaz respaldada por OpenTelemetry.

Requisitos previos

Actualiza las dependencias de tu proyecto para añadir el OpenTelemetry proveedor, tal y como se muestra en el siguiente fragmento de código de Gradle. Puedes navegar hasta el *X.Y.Z* enlace para ver la última versión disponible.

```
dependencies {
    implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
    implementation(platform("io.opentelemetry.instrumentation:opentelemetry-
instrumentation-bom:X.Y.Z"))
    implementation("aws.smithy.kotlin:telemetry-provider-otel")

    // OPTIONAL: If you use log4j, the following entry enables the ability to export
 logs through OTel.
    runtimeOnly("io.opentelemetry.instrumentation:opentelemetry-log4j-appender-2.17")
}
```

Configuración del SDK

El siguiente código configura un cliente de servicio mediante el proveedor de OpenTelemetry telemetría.

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.smithy.kotlin.runtime.telemetry.otel.OpenTelemetryProvider
import io.opentelemetry.api.GlobalOpenTelemetry
import kotlinx.coroutines.runBlocking
```

```
fun main() = runBlocking {
    val otelProvider = OpenTelemetryProvider(GlobalOpenTelemetry.get())

    S3Client.fromEnvironment().use { s3 ->
        telemetryProvider = otelProvider
        …
    }
}
```

> **Note**
>
> El análisis de cómo configurar el OpenTelemetry SDK queda fuera del ámbito de esta guía. La documentación de OpenTelemetry Java contiene información de configuración sobre los distintos enfoques: de forma manual, automática mediante el agente Java o mediante el recopilador (opcional).

Recursos

Los siguientes recursos están disponibles para ayudarle a empezar OpenTelemetry.

- AWS Distro for OpenTelemetry: página principal de AWS OTeL Distro
- aws-otel-java-instrumentation- Biblioteca de AWS instrumentación de Distro para Java OpenTelemetry
- aws-otel-lambda- capas OpenTelemetry Lambda AWS gestionadas
- aws-otel-collector- AWS Distro para Collector OpenTelemetry
- AWS Mejores prácticas de observabilidad: mejores prácticas generales para la observabilidad específicas de AWS

Anule la configuración del cliente del servicio

Una vez creado un cliente de servicio, este utiliza una configuración fija para todas las operaciones. Sin embargo, a veces es posible que necesite anular la configuración para una o más operaciones específicas.

Cada cliente de servicio tiene una `withConfig` extensión para que pueda modificar una copia de la configuración existente. La `withConfig` extensión devuelve un nuevo cliente de servicio con una

configuración modificada. El cliente original existe de forma independiente y utiliza su configuración original.

El siguiente ejemplo muestra la creación de una `S3Client` instancia que llama a dos operaciones.

```
val s3 = S3Client.fromEnvironment {
    logMode = LogMode.LogRequest
    region = "us-west-2"
    // ...other configuration settings...
}

s3.listBuckets { ... }
s3.listObjectsV2 { ... }
```

En el siguiente fragmento se muestra cómo anular la configuración de una sola operación. `listObjectV2`

```
s3.withConfig {
    region = "eu-central-1"
}.use { overriddenS3 ->
    overriddenS3.listObjectsV2 { ... }
}
```

Las llamadas a la operación del `s3` cliente utilizan la configuración original que se especificó cuando se creó el cliente. Su configuración incluye el registro de solicitudes y `us-west-2 region` para la región.

La `listObjectsV2` invocación en el `overriddenS3` cliente utiliza la misma configuración que en el `s3` cliente original, excepto en la región, que es ahora`eu-central-1`.

Ciclo de vida de un cliente anulado

En el ejemplo anterior, el `s3` cliente y el `overriddenS3` cliente son independientes entre sí. Las operaciones se pueden invocar en cualquiera de los clientes mientras permanezcan abiertos. Cada uno usa una configuración diferente, pero puede compartir los recursos subyacentes (como un HTTP motor) a menos que también se anulen.

Cierra un cliente con una configuración anulada y el cliente original por separado. Puede cerrar un cliente con una configuración anulada antes o después de cerrar su cliente original. A menos que necesite utilizar un cliente con una configuración anulada durante mucho tiempo, le recomendamos

que incluya este método en su ciclo de vida. `use` El `use` método garantiza que el cliente esté cerrado en caso de que se produzcan excepciones.

Recursos compartidos entre clientes

Al crear un cliente de servicio mediante el uso de`withConfig`, es posible que comparta recursos con el cliente original. Por el contrario, cuando se crea un cliente utilizando fromEnvironmento se configura de forma explícita, el cliente utiliza recursos independientes. Los recursos, como HTTP los motores y los proveedores de credenciales, se comparten a menos que se anulen en el `withConfig` bloque.

Como el ciclo de vida de cada cliente es independiente, los recursos compartidos permanecen abiertos y utilizables hasta que se cierre el último cliente. Por lo tanto, es importante que cierre los clientes de servicio anulados cuando ya no los necesite. Esto evita que los recursos compartidos permanezcan abiertos y consuman recursos del sistema, como la memoria, la conexión y CPU los ciclos.

El siguiente ejemplo muestra los recursos compartidos e independientes.

Los `overriddenS3` clientes `s3` y comparten la misma instancia del proveedor de credenciales, incluida su configuración de almacenamiento en caché. Las llamadas realizadas mediante credenciales `overriddenS3` reutilizan si el valor almacenado en caché sigue siendo el mismo que el de las llamadas realizadas por el `s3` cliente.

El HTTP motor no se comparte entre los dos clientes. Cada cliente tiene un HTTP motor independiente porque se anuló en la `withConfig` llamada.

```
val s3 = S3Client.fromEnvironment {
    region = "us-west-2"
    credentialsProvider = CachedCredentialsProvider(CredentialsProviderChain(...))
    httpClientEngine = OkHttpEngine { ... }
}

s3.listBuckets { ... }

s3.withConfig {
    httpClientEngine = CrtHttpEngine { ... }
}.use { overriddenS3 ->
    overriddenS3.listObjectsV2 { ... }
}
```

Utilizar SDK

En esta sección se proporciona la información básica necesaria para utilizar el AWS SDK para Kotlin.

Temas

Hacer solicitudes

Utilice un cliente de servicio para realizar solicitudes a un Servicio de AWS. AWS SDK para Kotlin Proporciona idiomas específicos de dominio (DSLs) que siguen un patrón de creación de tipos seguros para crear solicitudes. También se puede acceder a las estructuras anidadas de las solicitudes a través de sus. DSLs

El siguiente ejemplo muestra cómo crear una entrada de operación de Amazon createTableDynamoDB:

```
val ddb = DynamoDbClient.fromEnvironment()

val req = CreateTableRequest {
    tableName = name
    keySchema = listOf(
        KeySchemaElement {
            attributeName = "year"
            keyType = KeyType.Hash
        },
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }
```

```
    )

    attributeDefinitions = listOf(
        AttributeDefinition {
            attributeName = "year"
            attributeType = ScalarAttributeType.N
        },
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }
    )

    // You can configure the `provisionedThroughput` member
    // by using the `ProvisionedThroughput.Builder` directly:
    provisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }
}

val resp = ddb.createTable(req)
```

Sobrecargas de la interfaz de servicio DSL

Cada operación no relacionada con la transmisión en la interfaz del cliente del servicio está DSL sobrecargada, por lo que no es necesario crear una solicitud independiente.

Ejemplo de creación de un depósito de Amazon Simple Storage Service (Amazon S3) con la función de sobrecarga:

```
s3Client.createBucket {    // this: CreateBucketRequest.Builder
    bucket = newBucketName
}
```

Equivale a:

```
val request = CreateBucketRequest {    // this: CreateBucketRequest.Builder
    bucket = newBucketName
}

s3client.createBucket(request)
```

Solicitudes sin entradas obligatorias

Se puede llamar a las operaciones que no tienen entradas obligatorias sin tener que pasar un objeto de solicitud. Esto suele ser posible con operaciones de tipo lista, como la operación Amazon S3 `listBuckets`API.

Por ejemplo, las tres afirmaciones siguientes son equivalentes:

```
s3Client.listBuckets(ListBucketsRequest {
  // Construct the request object directly.
})
s3Client.listBuckets {
  // DSL builder without explicitly setting any arguments.
}
s3Client.listBuckets()
```

Corrutinas

AWS SDK para Kotlin Es asíncrono de forma predeterminada. El de SDK Kotlin usa `suspend` funciones para todas las operaciones, que deben ser llamadas desde una corrutina.

Para obtener una guía más detallada sobre las corrutinas, consulta la documentación oficial de Kotlin.

Hacer solicitudes simultáneas

El generador de corrutinas asíncronas se puede utilizar para lanzar solicitudes simultáneas cuando le interesen los resultados. `async`devuelve un Deferred, que representa un futuro ligero y sin bloqueos que representa la promesa de ofrecer un resultado más adelante.

Si no te importan los resultados (solo que se haya completado una operación), puedes usar el generador de corrutinas de lanzamiento. `launches` conceptualmente similar a. `async` La diferencia es que launch devuelve un Job y no contiene ningún valor resultante, mientras que `async` devuelve un`Deferred`.

El siguiente es un ejemplo de cómo realizar solicitudes simultáneas a Amazon S3 mediante la headObjectoperación para obtener el tamaño del contenido de dos claves:

```
import kotlinx.coroutines.async
```

```
import kotlinx.coroutines.runBlocking
import kotlin.system.measureTimeMillis
import aws.sdk.kotlin.services.s3.S3Client


fun main(): Unit = runBlocking {

    val s3 = S3Client { region = "us-east-2" }

    val myBucket = "<your-bucket-name-here>"
    val key1 = "<your-object-key-here>"
    val key2 = "<your-second-object-key-here>"

    val resp1 = async {
        s3.headObject{
            bucket = myBucket
            key = key1
        }
    }

    val resp2 = async {
        s3.headObject{
            bucket = myBucket
            key = key2
        }
    }


    val elapsed = measureTimeMillis {
        val totalContentSize = resp1.await().contentLength +
 resp2.await().contentLength
        println("content length of $key1 + $key2 = $totalContentSize")
    }

    println("requests completed in $elapsed ms")

}
```

Realizar solicitudes de bloqueo

Para realizar llamadas de servicio desde un código existente que no usa corrutinas e implementa un modelo de subprocesos diferente, puedes usar el generador de runBlockingcorrutinas. Un ejemplo de un modelo de subprocesamiento diferente es el uso del enfoque tradicional de ejecutores y

futuros de Java. Es posible que necesites usar este enfoque si vas a combinar código o bibliotecas de Java y Kotlin.

Como su nombre indica, este `runBlocking` generador lanza una nueva corrutina y bloquea el hilo actual hasta que se complete.

> **Warning**
>
> `runBlocking`por lo general, no debe usarse desde una corrutina. Está diseñado para vincular el código de bloqueo normal con las bibliotecas que están escritas en forma suspensiva (por ejemplo, en las funciones principales y las pruebas).

Operaciones de streaming

En el AWS SDK para Kotlin, los datos binarios (flujos) se representan como un `ByteStream`tipo, que es un flujo abstracto de bytes de solo lectura.

Respuestas de transmisión

Las respuestas con un flujo binario (como la operación Amazon Simple Storage Service (Amazon GetObjectAPIS3)) se gestionan de forma diferente a la de otros métodos. Estos métodos utilizan una función lambda que gestiona la respuesta en lugar de devolverla directamente. Esto limita el alcance de la respuesta a la función y simplifica la administración del ciclo de vida, tanto para la persona que llama como para el tiempo de ejecución. SDK

Una vez que la función lambda regresa, se liberan todos los recursos, como la HTTP conexión subyacente. (No se `ByteStream` debe acceder a él después de que la lambda regrese y no se debe pasar del cierre). El resultado de la llamada es lo que devuelva la lambda.

El siguiente ejemplo de código muestra que la getObjectfunción recibe un parámetro lambda, que gestiona la respuesta.

```
val s3Client = S3Client.fromEnvironment()
val req = GetObjectRequest { ... }

val path = Paths.get("/tmp/download.txt")

// S3Client.getObject has the following signature:
```

```
// suspend fun <T> getObject(input: GetObjectRequest, block: suspend
 (GetObjectResponse) -> T): T

val contentSize = s3Client.getObject(req) { resp ->
    // resp is valid until the end of the block.
    // Do not attempt to store or process the stream after the block returns.

    // resp.body is of type ByteStream.
    val rc = resp.body?.writeToFile(path)
    rc
}
println("wrote $contentSize bytes to $path")
```

El `ByteStream` tipo tiene las siguientes extensiones para las formas habituales de consumirlo:

- `ByteStream.writeToFile(file: File): Long`
- `ByteStream.writeToFile(path: Path): Long`
- `ByteStream.toByteArray(): ByteArray`
- `ByteStream.decodeToString(): String`

Todas ellas están definidas en el `aws.smithy.kotlin.runtime.content` paquete.

Solicitudes de streaming

Para suministrar un`ByteStream`, también hay varios métodos prácticos, incluidos los siguientes:

- `ByteStream.fromFile(file: File)`
- `File.asByteStream(): ByteStream`
- `Path.asByteStream(): ByteStream`
- `ByteStream.fromBytes(bytes: ByteArray)`
- `ByteStream.fromString(str: String)`

Todos ellos están definidos en el `aws.smithy.kotlin.runtime.content` paquete.

El siguiente ejemplo de código muestra el uso de métodos `ByteStream` prácticos que proporcionan la propiedad body en la creación de un PutObjectRequest:

```
val req = PutObjectRequest {
```

```
    ...
    body = ByteStream.fromFile(file)
    // body = ByteStream.fromBytes(byteArray)
    // body = ByteStream.fromString("string")
    // etc
}
```

Paginación

Muchas AWS operaciones devuelven resultados paginados cuando la carga útil es demasiado grande como para devolverlos en una sola respuesta. AWS SDK para Kotlin Incluye extensiones de la interfaz del cliente de servicio que paginan automáticamente los resultados por usted. Solo tienes que escribir el código que procesa los resultados.

La paginación se presenta como un flujo <T>para que puedas aprovechar las transformaciones idiomáticas de Kotlin para colecciones asíncronas (como, y). `map` `filter` `take` Las excepciones son transparentes, lo que hace que la gestión de errores parezca una API llamada normal, y la cancelación se basa en la cancelación cooperativa general de las corrutinas. Para obtener más información, consulta los flujos y las excepciones de flujo en la guía oficial.

> **Note**
>
> Los siguientes ejemplos utilizan Amazon S3. Sin embargo, los conceptos son los mismos para cualquier servicio que tenga uno o más paginadosAPIs. Todas las extensiones de paginación se definen en el `aws.sdk.kotlin.<service>.paginators` paquete (por ejemplo,). `aws.sdk.kotlin.dynamodb.paginators`

El siguiente ejemplo de código muestra cómo se puede procesar la respuesta paginada de la llamada a la función listObjectsV2Paginated.

Importaciones

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.paginators.listObjectsV2Paginated
import kotlinx.coroutines.flow.*
```

Código

```
val s3 = S3Client.fromEnvironment()
val req = ListObjectsV2Request {
    bucket = "<my-bucket>"
    maxKeys = 1
}

s3.listObjectsV2Paginated(req)  // Flow<ListObjectsV2Response>
    .transform { it.contents?.forEach { obj -> emit(obj) } }
    .collect { obj ->
        println("key: ${obj.key}; size: ${obj.size}")
    }
```

Esperadores

Los camareros son una abstracción del lado del cliente que se utiliza para sondear un recurso hasta que se alcance el estado deseado o hasta que se determine que el recurso no entrará en el estado deseado. Esta es una tarea habitual cuando se trabaja con servicios que finalmente son coherentes, como Amazon Simple Storage Service (Amazon S3), o servicios que crean recursos de forma asíncrona, como Amazon. EC2

Escribir la lógica para sondear continuamente el estado de un recurso puede resultar engorroso y propenso a errores. El objetivo de los camareros es sacar esta responsabilidad del código de cliente y llevarla a una empresa que tenga un conocimiento profundo de los AWS SDK para Kotlin aspectos relacionados con el tiempo de la operación. AWS

> **Note**
>
> Los siguientes ejemplos utilizan Amazon S3. Sin embargo, los conceptos son los mismos para todos aquellos en los Servicio de AWS que se hayan definido uno o más camareros. Todas las extensiones se definen en el `aws.sdk.kotlin.<service>.waiters` paquete (por ejemplo,`aws.sdk.kotlin.dynamodb.waiters`). También siguen una convención de nomenclatura estándar (`waitUntil<Condition>`).

El siguiente ejemplo de código muestra el uso de una función de camarero que permite evitar escribir la lógica de las votaciones.

Importaciones

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.waiters.waitUntilBucketExists
```

Código

```
val s3 = S3Client.fromEnvironment()

// This initiates creating an S3 bucket and potentially returns before the bucket
 exists.
s3.createBucket { bucket = "my-bucket" }

// When this function returns, the bucket either exists or an exception
// is thrown.
s3.waitUntilBucketExists { bucket = "my-bucket" }

// The bucket now exists.
```

> **Note**
>
> Cada método de espera devuelve una `Outcome` instancia que se puede utilizar para obtener la respuesta final que corresponde a alcanzar la condición deseada. El resultado también contiene detalles adicionales, como el número de intentos realizados para alcanzar el estado deseado.

Gestión de errores

Comprender cómo y cuándo se AWS SDK para Kotlin producen excepciones es importante para crear aplicaciones de alta calidad utilizandoSDK. En las siguientes secciones se describen los diferentes casos de excepciones que genera SDK y cómo gestionarlos adecuadamente.

Excepciones de servicio

La excepción más común es `AwsServiceException` aquella de la que se heredan todas las excepciones específicas de un servicio (por ejemplo`S3Exception`). Esta excepción representa una respuesta de error de un servicio de Servicio de AWS. Por ejemplo, si intentas cerrar una EC2 instancia de Amazon que no existe, Amazon EC2 devuelve una respuesta de error. Los detalles de la respuesta al error se incluyen en el mensaje `AwsServiceException` que aparece.

Si encuentras un mensaje`AwsServiceException`, significa que tu solicitud se envió correctamente Servicio de AWS , pero no se pudo procesar. Esto puede ser debido a errores en los parámetros de la solicitud o a problemas en el servicio.

Excepciones de clientes

`ClientException`indica que se ha producido un problema en el código del AWS SDK para Kotlin cliente, ya sea al intentar enviar una solicitud AWS o al intentar analizar una respuesta desde él AWS. Por lo general, A `ClientException` es más grave que a `AwsServiceException` e indica que un problema importante es impedir que el cliente procese las llamadas de servicio a Servicios de AWS. Por ejemplo, AWS SDK para Kotlin lanza un `ClientException` si no puede analizar la respuesta de un servicio.

Metadatos de error

Cada excepción de servicio y excepción de cliente tiene `sdkErrorMetadata` esta propiedad. Se trata de una bolsa de propiedades mecanografiada que se puede utilizar para recuperar detalles adicionales sobre el error.

Existen varias extensiones predefinidas para el `AwsErrorMetadata` tipo directamente, incluidas, entre otras, las siguientes:

- `sdkErrorMetadata.requestId`— el identificador único de la solicitud
- `sdkErrorMetadata.errorMessage`— el mensaje legible por humanos (normalmente coincide con el`Exception.message`, pero puede contener más información si el servicio desconoce la excepción)
- `sdkErrorMetadata.protocolResponse`— La respuesta del protocolo sin procesar

En el siguiente ejemplo, se muestra el acceso a los metadatos del error.

```
try {
    s3Client.listBuckets { ... }
} catch (ex: S3Exception) {
    val awsRequestId = ex.sdkErrorMetadata.requestId
    val httpResp = ex.sdkErrorMetadata.protocolResponse as? HttpResponse

    println("requestId was: $awsRequestId")
    println("http status code was: ${httpResp?.status}")
}
```

Solicitudes de prefirma

Puede prefirmar las solicitudes de algunas AWS API operaciones para que otra persona que llame pueda utilizar la solicitud más adelante sin tener que presentar sus propias credenciales.

Por ejemplo, supongamos que Alice tiene acceso a un objeto del Amazon Simple Storage Service (Amazon S3) y quiere compartir temporalmente el acceso al objeto con Bob. Alice puede generar una `GetObject` solicitud prefirmada para compartirla con Bob, de forma que pueda descargar el objeto sin necesidad de acceder a las credenciales de Alice.

Conceptos básicos de prefirma

SDKPara Kotlin, se proporcionan métodos de extensión a los clientes del servicio para prefirmar las solicitudes. Todas las solicitudes prefirmadas requieren una duración que representa el tiempo de validez de la solicitud firmada. Una vez finalizada la duración, la solicitud prefirmada caduca y, si se ejecuta, genera un error de autenticación.

En el siguiente código se muestra un ejemplo que crea una `GetObject` solicitud prefirmada para Amazon S3. La solicitud es válida durante 24 horas después de su creación.

```
val s3 = S3Client.fromEnvironment()

val unsignedRequest = GetObjectRequest {
    bucket = "foo"
    key = "bar"
}

val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)
```

El método `presignGetObject`de extensión devuelve un `HttpRequest`objeto. El objeto de solicitud contiene el lugar prefirmado en el URL que se puede invocar la operación. Otra persona que llame puede usar la solicitud URL (o toda la solicitud) en una base de código o entorno de lenguaje de programación diferente.

Después de crear la solicitud prefirmada, utilice un HTTP cliente para invocarla. La API invocación de una HTTP GET solicitud depende del cliente. HTTP En el siguiente ejemplo, se utiliza el método Kotlin `URL.readText`.

```
val objectContents = URL(presignedRequest.url.toString()).readText()
```

```
println(objectContents)
```

Configuración avanzada de prefirma

En laSDK, cada método que puede prefirmar las solicitudes tiene una sobrecarga que puede utilizar para proporcionar opciones de configuración avanzadas, como la implementación de un firmante específico o parámetros de firma detallados.

El siguiente ejemplo muestra una `GetObject` solicitud de Amazon S3 que usa la variante de CRT firmante y especifica una fecha de firma futura.

```
val s3 = S3Client.fromEnvironment()

val unsignedRequest = GetObjectRequest {
    bucket = "foo"
    key = "bar"
}

val presignedRequest = s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
    signingDate = Instant.now() + 24.hours
    expiresAfter = 8.hours
}
```

La solicitud prefirmada devuelta tiene una fecha de reenvío de 24 horas y no es válida antes de esa fecha. Expira 8 horas después.

Prefirma POST y solicitudes PUT

Muchas operaciones que se pueden prefirmar solo requieren una URL y deben ejecutarse como solicitudes. HTTP GET Sin embargo, algunas operaciones ocupan un cuerpo y, en algunos casos, deben ejecutarse como una HTTP PUT solicitud HTTP POST o junto con encabezados. Prefirmar estas solicitudes es idéntico a prefirmar GET las solicitudes, pero invocar la solicitud prefirmada es más complicado.

A continuación, se muestra un ejemplo de prefirma de una solicitud de S3: `PutObject`

```
val s3 = S3Client.fromEnvironment()

val unsignedRequest = PutObjectRequest {
    bucket = "foo"
```

```
    key = "bar"
}

val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)
```

El valor devuelto `HttpRequest` tiene un valor de método de `HttpMethod.PUT` e incluye encabezados URL y que deben incluirse en la futura invocación de la HTTP solicitud. Puedes pasar esta solicitud a una persona que llame para que la ejecute en una base de código o entorno de lenguaje de programación diferente.

Después de crear el prefirmado POST o la PUT solicitud, usa un HTTP cliente para invocar una solicitud. La API invocación de una PUT solicitud POST o URL depende del HTTP cliente utilizado. El siguiente ejemplo utiliza un OkHttp HTTPcliente e incluye un cuerpo que contiene`Hello world`.

```
val putRequest = Request
    .Builder()
    .url(presignedRequest.url.toString())
    .apply {
        presignedRequest.headers.forEach { key, values ->
            header(key, values.joinToString(", "))
        }
    }
    .put("Hello world".toRequestBody())
    .build()

val response = okHttp.newCall(putRequest).execute()
```

Operaciones que SDK pueden prefirmar

Actualmente, SDK for Kotlin admite la predefinición de las siguientes API operaciones que deben invocarse con el método asociado. HTTP

Servicio de AWS	Operación	SDKmétodo de extensión	Utilice el HTTP método
Amazon S3	GetObject	presignGetObject	HTTP GET
Amazon S3	PutObject	presignPutObject	HTTP PUT
Amazon S3	UploadPart	presignUploadPart	HTTP PUT

Servicio de AWS	Operación	SDKmétodo de extensión	Utilice el HTTP método
AWS Security Token Service	GetCallerIdentity	presignGetCallerIdentidad	HTTP POST
Amazon Polly	SynthesizeSpeech	presignSynthesizeSpeech	HTTP POST

Solución de problemas de FAQs

A medida que lo utilice AWS SDK para Kotlin en sus aplicaciones, es posible que encuentre algunos de los problemas enumerados en este tema. Utilice las siguientes sugerencias para ayudar a descubrir la causa raíz y resolver el error.

¿Cómo soluciono los problemas de «conexión cerrada»?

Es posible que se produzcan problemas de «conexión cerrada» como excepciones, como alguno de los siguientes tipos:

- `IOException: unexpected end of stream on <URL>`
- `EOFException: \n not found: limit=0`
- `HttpException: AWS_ERROR_HTTP_CONNECTION_CLOSED: The connection has closed or is closing.; crtErrorCode=2058; HttpErrorCode(CONNECTION_CLOSED)`

Estas excepciones indican que una TCP conexión desde un servicio se cerró o restableció inesperadamente. SDK Es posible que el anfitrión, el AWS servicio o un intermediario, como una NAT puerta de enlace, un proxy o un equilibrador de carga, hayan cerrado la conexión.

Estos tipos de excepciones se vuelven a intentar automáticamente, pero es posible que sigan apareciendo en SDK los registros, según la configuración de registro. Si la excepción se incluye en el código, eso indica que la estrategia de reintentos activa ha agotado los límites configurados, como el número máximo de intentos o el número de reintentos. Consulta la the section called “Reintentos” sección de esta guía para obtener más información sobre las estrategias de reintentos. Consulte también el the section called “¿Por qué se hacen excepciones antes de alcanzar el máximo de intentos?” tema?.

¿Por qué se hacen excepciones antes de alcanzar el máximo de intentos?

A veces, es posible que veas excepciones que esperabas que se volvieran a intentar, pero que en su lugar se han lanzado. En estas situaciones, los siguientes pasos pueden ayudar a resolver el problema.

- Compruebe que la excepción se puede volver a intentar. Algunas excepciones no se pueden volver a intentar, como las que indican una solicitud de servicio con un formato incorrecto, falta de permisos o recursos inexistentes, por ejemplo. SDKNo vuelve a intentar automáticamente este tipo de excepciones. Si detectas una excepción heredada de`SdkBaseException`, puedes comprobar la propiedad booleana `SdkBaseException.sdkErrorMetadata.isRetryable` para comprobar si SDK ha determinado que la excepción se puede volver a intentar.
- Comprueba que la excepción esté incluida en tu código. Algunas excepciones aparecen en los mensajes de registro como información, pero en realidad no se incluyen en el código. Por ejemplo, las excepciones que se pueden volver a intentar, como los errores de limitación, pueden registrarse, ya que funcionan SDK automáticamente durante varios ciclos. backoff-and-retry La invocación de una SDK operación genera una excepción solo si no se ha gestionado con los ajustes de reintento configurados.
- Compruebe los ajustes de reintento configurados. Consulte la the section called “Reintentos” sección de esta guía para obtener más información sobre las estrategias y políticas de reintentos. Asegúrese de que el código utilice la configuración esperada o los valores predeterminados automáticos.
- Considera la posibilidad de ajustar la configuración de reintentos. Tras comprobar los elementos anteriores, pero el problema no se haya resuelto, podría plantearse la posibilidad de ajustar la configuración de los reintentos.
  - Aumente el número máximo de intentos. De forma predeterminada, el número máximo de intentos de una operación es 3. Si considera que esto no es suficiente y se siguen produciendo excepciones con la configuración predeterminada, considere la posibilidad de aumentar la `retryStrategy.maxAttempts` propiedad en la configuración del cliente. Para obtener más información, consulta the section called “Número máximo de intentos”.
  - Aumente la configuración de retardo. Es posible que algunas excepciones se vuelvan a intentar con demasiada rapidez antes de que la afección subyacente haya tenido la oportunidad de resolverse. Si sospecha que es así, considere la posibilidad de aumentar las `retryStrategy.delayProvider.maxBackoff` propiedades `retryStrategy.delayProvider.initialDelay` o en la configuración de su cliente. Para obtener más información, consulta the section called “Retrasos y retrasos”.

- Desactive el modo de disyuntor. De forma predeterminada, SDK mantiene un conjunto de fichas para cada cliente de servicio. Cuando se SDK intenta realizar una solicitud y se produce un error con una excepción que se puede volver a intentar, el recuento de fichas disminuye; cuando la solicitud es correcta, el recuento de fichas se incrementa.

  De forma predeterminada, si este depósito de fichas llega a las 0 fichas restantes, el circuito se interrumpe. Cuando se interrumpe el circuito, se SDK desactivan los reintentos y cualquier solicitud actual o posterior que falle en el primer intento genera inmediatamente una excepción. Los SDK reintentos de reactivación tras los intentos iniciales satisfactorios devuelven suficiente capacidad al depósito de fichas. Este comportamiento es intencional y está diseñado para evitar una gran cantidad de reintentos durante las interrupciones del servicio y la recuperación del servicio.

  Si prefiere volver a intentarlo hasta alcanzar SDK el máximo de intentos configurados, considere la posibilidad de deshabilitar el modo de disyuntor estableciendo la `retryStrategy.tokenBucket.useCircuitBreakerMode` propiedad en false en la configuración de su cliente. Con esta propiedad establecida en false, el SDK cliente espera a que el depósito de fichas alcance la capacidad suficiente en lugar de abandonar los reintentos posteriores, lo que podría provocar una excepción cuando queden 0 fichas.

¿Cómo puedo corregir **`NoSuchMethodError`** o? NoClassDefFoundError

Se SDK basa en dependencias diversas AWS y de terceros para funcionar correctamente. Si las dependencias esperadas no están presentes en el tiempo de ejecución o son una versión inesperada, es posible que veas la excepción del `NoSuchMethodError` tiempo de ejecución.

Los conflictos de dependencia suelen clasificarse en dos categorías: los conflictos de dependencia de SDK /Smithy y los conflictos de dependencia de terceros.

Cuando compilas una aplicación de Kotlin, normalmente administras las dependencias con Gradle. Añadir una dependencia de un cliente de SDK servicio a tu aplicación debería resolver e incluir automáticamente todas las dependencias transitivas. Si la aplicación tiene otras dependencias, es posible que entren en conflicto con las que necesita SDK (por ejemplo, si OkHttp se trata de un HTTP cliente de uso común del que depende). SDK

Para resolver estos problemas, es posible que tengas que resolver de forma explícita una versión de dependencia específica o esconder dependencias en un espacio de nombres local para evitar

el conflicto. La resolución de dependencias de Gradle es un tema complejo que se analiza en las siguientes secciones del manual del usuario de Grade.

- Comprensión de la resolución de dependencias
- Restricciones de dependencia y resolución de conflictos
- Alineación de las versiones de dependencia

SDK/Conflictos de dependencia de Smithy

En general, los módulos dependen SDK de otros SDK módulos con el mismo número de versión. Por ejemplo, `aws.sdk.kotlin:s3:1.2.3` depende de`aws.sdk.kotlin:aws-http:1.2.3`, de qué de`aws.sdk.kotlin:aws-core:1.2.3`pende, etc.

Además, SDK los módulos también se basan en versiones específicas y unificadas de los módulos Smithy. Estos números de versión de Smithy no están sincronizados con los números de SDK versión, pero deben coincidir con la versión esperada por el. SDK Por ejemplo, `aws.sdk.kotlin:s3:1.2.3` puede depender de`aws.smithy.kotlin:serde:1.1.1`, de qué de`aws.smithy.kotlin:runtime-core:1.1.1`pende, etc.

Si alguno de estos números de versión no coincide, es posible que se produzcan conflictos de dependencia. Asegúrese de actualizar todas sus SDK dependencias al mismo tiempo y de actualizar también todas las dependencias explícitas de Smithy al mismo tiempo. Considera usar nuestro catálogo de versiones de Gradle para mantener las versiones sincronizadas y eliminar las conjeturas entre las versiones y las de Smithy. SDK Consulta el the section called “Cree archivos de compilación del proyecto” tema para obtener más información y ejemplos.

Además, tenga en cuenta que los pequeños errores de versión de los módulos de SDK /Smithy pueden contener cambios importantes, tal y como se describe en la política de control SDK de versiones de la misma. Al actualizar entre versiones secundarias, procure examinar los registros de cambios y validar minuciosamente el comportamiento en tiempo de ejecución.

Veo un formulario **`NoClassDefFoundErrorokhttp3/coroutines/ExecuteAsyncKt`**

Si ve este error, lo más probable es que no haya configurado su cliente de servicio para usar el`0kHttp4Engine`. Revisa la documentación sobre cómo configurar Gradle y usarlo `0kHttp4Engine` en tu código.

Trabaje con el Servicios de AWS uso del AWS SDK para Kotlin

Este capítulo contiene información sobre cómo trabajar Servicios de AWS con Kotlin. SDK

Contenido

Trabaje con Amazon S3 mediante el AWS SDK para Kotlin

La interfaz principal del Amazon Simple Storage Service para Kotlin SDK es S3Client. `S3Client`Utilícelo como otros clientes de servicio del SDK para realizar solicitudes a Amazon S3.

Los recursos que le ayudarán a usar Kotlin SDK con S3 son:

- la SDK APIreferencia de Kotlin para S3.
- la guía del usuario y la APIreferencia del servicio S3.

En los siguientes temas se presentan ejemplos de códigos guiados para algunos Kotlin SDK APIs que funcionan con S3.

Temas

- Sumas de comprobación de Amazon S3 con AWS SDK para KotlinAWS SDK for Java
- Trabaje con los puntos de acceso multirregionales de Amazon S3 mediante el SDK para Kotlin

Sumas de comprobación de Amazon S3 con AWS SDK para KotlinAWS 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 -32C. CRC Tiene la flexibilidad de elegir el algoritmo que mejor se adapte a sus necesidades y dejar que él calcule la suma de verificación. SDK 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

Para cargar objetos a Amazon S3 SDK para Kotlin, utilice la `putObject`función con un parámetro de solicitud. El tipo de datos de la solicitud indica la propiedad `checksumAlgorithm` para que se

pueda calcular la suma de comprobación. 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 verificación de CRC -32. Cuando SDK envía la solicitud, calcula la suma de control CRC -32 y carga el objeto. Amazon S3 almacena la suma de comprobación en el objeto.

```
val request = PutObjectRequest {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumAlgorithm = ChecksumAlgorithm.CRC32
}
```

Si la suma de comprobación que SDK calcula no coincide con la suma de comprobación que Amazon S3 calcula al recibir la solicitud, se devuelve un error.

Utilizar un valor de suma de comprobación calculado previamente

Un valor de suma de comprobación precalculado que se proporciona con la solicitud desactiva el cálculo automático por parte de y, en su lugar, utiliza el SDK valor proporcionado.

En el siguiente ejemplo, se muestra una solicitud con una suma de comprobación precalculada de -256. SHA

```
val request = PutObjectRequest {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    body = ByteStream.fromFile(File("file_to_upload.txt"))
    checksumAlgorithm = ChecksumAlgorithm.SHA256
    checksumSha256 = "cfb6d06da6e6f51c22ae3e549e33959dbb754db75a93665b8b579605464ce299"
}
```

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. Debe especificar el algoritmo de suma de comprobación en la `CreateMultipartUpload` solicitud y en cada solicitud. `UploadPart` Por último, debe especificar la suma de comprobación de cada parte en

`CompleteMultipartUpload`. En el siguiente ejemplo, se muestra cómo crear una carga multiparte con el algoritmo de suma de comprobación especificado.

```
val multipartUpload = s3.createMultipartUpload {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumAlgorithm = ChecksumAlgorithm.Sha1
}

val partFilesToUpload = listOf("data-part1.csv", "data-part2.csv", "data-part3.csv")

val completedParts = partFilesToUpload
    .mapIndexed { i, fileName ->
        val uploadPartResponse = s3.uploadPart {
            bucket = "amzn-s3-demo-bucket"
            key = "key"
            body = ByteStream.fromFile(File(fileName))
            uploadId = multipartUpload.uploadId
            partNumber = i + 1 // Part numbers begin at 1.
            checksumAlgorithm = ChecksumAlgorithm.Sha1
        }

        CompletedPart {
            eTag = uploadPartResponse.eTag
            partNumber = i + 1
            checksumSha1 = uploadPartResponse.checksumSha1
        }
    }

s3.completeMultipartUpload {
    uploadId = multipartUpload.uploadId
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    multipartUpload {
        parts = completedParts
    }
}
```

Descargar un objeto

Cuando utilizas el getObjectmétodo para descargar un objeto, valida SDK automáticamente la suma de verificación . `enabled` cuando la `checksumMode` propiedad del generador para el `GetObjectRequest` está establecida en`ChecksumMode.Enabled`.

La solicitud del siguiente fragmento indica que validen la SDK suma de verificación de la respuesta calculándola y comparando los valores.

```
val request = GetObjectRequest {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumMode = ChecksumMode.Enabled
}
```

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: SDK

1. CRC-32°C
2. CRC-32
3. SHA-1
4. SHA-256

Por ejemplo, si una respuesta contiene sumas de verificación de CRC -32 y de SHA -256, solo se valida la suma de verificación CRC -32.

Validación asíncrona

Como SDK for Kotlin utiliza respuestas de streaming cuando descarga un objeto de Amazon S3, la suma de comprobación se calculará a medida que consumas el objeto. Por lo tanto, debe usar el objeto para validar la suma de comprobación.

En el siguiente ejemplo se muestra cómo validar una suma de comprobación consumiendo por completo la respuesta.

```
val request = GetObjectRequest {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumMode = checksumMode.Enabled
}

val response = s3Client.getObject(request) {
    println(response.body?.decodeToString()) // Fully consume the object.
```

```
    // The checksum is valid.
}
```

Por el contrario, el código del siguiente ejemplo no usa el objeto de ninguna manera, por lo que la suma de comprobación no se valida.

```
s3Client.getObject(request) {
    println("Got the object.")
}
```

Si la suma de comprobación calculada por el SDK no coincide con la suma de comprobación esperada enviada con la respuesta, arroja un. SDK `ChecksumMismatchException`

Trabaje con los puntos de acceso multirregionales de Amazon S3 mediante el SDK para Kotlin

Los puntos de acceso multirregión de Amazon S3 proporcionan un punto de conexión global que las aplicaciones pueden utilizar para satisfacer las solicitudes de los buckets de S3 ubicados en varias Regiones de AWS. Puede utilizar puntos de acceso multirregión para crear aplicaciones de multirregiones con la misma arquitectura utilizada en una sola región y, a continuación, ejecutar esas aplicaciones en cualquier parte del mundo.

La guía del usuario de Amazon S3 contiene más información básica sobre los puntos de acceso multirregionales.

Trabaje con puntos de acceso multirregionales

Para crear un punto de acceso multirregional, comience por especificar un segmento en cada AWS región en el que desee atender las solicitudes. El siguiente fragmento crea dos depósitos.

Creación de buckets

La siguiente función crea dos depósitos para que funcionen con el punto de acceso multirregional. Un depósito está en la región `us-east-1` y el otro en la región. `us-west-1`

La creación del cliente S3 que pasó como primer argumento se muestra en el primer ejemplo de abajothe section called “Uso de objetos de ”.

```
        suspend fun setUpTwoBuckets(
            s3: S3Client,
            bucketName1: String,
```

```
            bucketName2: String,
        ) {
            println("Create two buckets in different regions.")
            // The shared aws config file configures the default Region to be us-
east-1.
            s3.createBucket(
                CreateBucketRequest {
                    bucket = bucketName1
                },
            )
            s3.waitUntilBucketExists {
                bucket = bucketName1
            }
            println("  Bucket [$bucketName1] created.")

            // Override the S3Client to work with us-west-1 for the second bucket.
            s3.withConfig {
                region = "us-west-1"
            }.use { s3West ->
                s3West.createBucket(
                    CreateBucketRequest {
                        bucket = bucketName2
                        createBucketConfiguration = CreateBucketConfiguration {
                            locationConstraint = BucketLocationConstraint.UsWest1
                        }
                    },
                )
                s3West.waitUntilBucketExists {
                    bucket = bucketName2
                }
                println("  Bucket [$bucketName2] created.")
            }
        }
```

Utiliza el cliente de control S3 SDK de Kotlin para crear, eliminar y obtener información sobre los puntos de acceso multirregionales.

Agrega una dependencia al artefacto de control de S3, como se muestra en el siguiente fragmento. (Puede navegar hasta el *X.Y.Z* enlace para ver la última versión disponible).

```
...
implementation(platform("aws.sdk.kotlin:bom:X.Y.Z"))
implementation("aws.sdk.kotlin:s3control")
```

```
...
```

Configure el cliente de control S3 para que funcione Región de AWS `us-west-2` como se muestra en el siguiente código. Todas las operaciones del cliente de control de S3 deben dirigirse a la `us-west-2` región.

```
        suspend fun createS3ControlClient(): S3ControlClient {
            // Configure your S3ControlClient to send requests to US West (Oregon).
            val s3Control = S3ControlClient.fromEnvironment {
                region = "us-west-2"
            }
            return s3Control
        }
```

Utilice el cliente de control S3 para crear un punto de acceso multirregional especificando los nombres de los buckets (creados anteriormente), tal y como se muestra en el código siguiente.

```
    suspend fun createMrap(
        s3Control: S3ControlClient,
        accountIdParam: String,
        bucketName1: String,
        bucketName2: String,
        mrapName: String,
    ): String {
        println("Creating MRAP ...")
        val createMrapResponse: CreateMultiRegionAccessPointResponse =
            s3Control.createMultiRegionAccessPoint {
                accountId = accountIdParam
                clientToken = UUID.randomUUID().toString()
                details {
                    name = mrapName
                    regions = listOf(
                        Region {
                            bucket = bucketName1
                        },
                        Region {
                            bucket = bucketName2
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn
```

```
        // Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

Como la creación de un punto de acceso multirregional es una operación asíncrona, se utiliza el token que se recibe de la respuesta inmediata para comprobar el estado del proceso de creación. Una vez que la comprobación de estado muestre un mensaje de éxito, puede utilizar la `GetMultiRegionAccessPoint` operación para obtener el alias del punto de acceso multirregional. El alias es el último componente delARN, que se necesita para las operaciones a nivel de objeto.

Utilice un token para comprobar el estado

Utilice el `DescribeMultiRegionAccessPointOperation` para comprobar el estado de la última operación. Cuando el `requestStatus` valor pase a ser "SUCCEEDED«, podrá trabajar con el punto de acceso multirregional.

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
```

```
            },
        )

        var status: String? = describeResponse.asyncOperation?.requestStatus
        while (status != "SUCCEEDED") {
            delay(timeBetweenChecks)
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )
            status = describeResponse.asyncOperation?.requestStatus
            println(status)
        }
    }
```

Trabaje con objetos y puntos de acceso multirregionales

El cliente S3 se utiliza para trabajar con objetos en puntos de acceso multirregionales. Muchas de las operaciones que se realizan con objetos en depósitos se pueden realizar en puntos de acceso multirregionales. Para obtener más información y una lista completa de las operaciones, consulte Compatibilidad de los puntos de acceso multirregionales con las operaciones de S3.

Las operaciones con puntos de acceso multirregionales se firman con el algoritmo de firma asimétrico Sigv4 (SiGV4a). Support for Sigv4a AWS SDK para Kotlin actualmente requiere firmar con el CRT firmante, que es una dependencia independiente. Para configurar el soporte para SigV4a, añade las siguientes dependencias a tu proyecto. (Puedes navegar hasta el *X.Y.Z* enlace para ver la última versión disponible).

```
...
implementation(platform("aws.sdk.kotlin:bom:X.Y.Z"))
implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))

implementation("aws.smithy.kotlin:aws-signing-crt")
implementation("aws.smithy.kotlin:http-auth-aws")
implementation("aws.sdk.kotlin:s3")
...
```

Tras añadir las dependencias, configure el cliente S3 para que utilice el algoritmo de firma SigV4a, tal como se muestra en el código siguiente.

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric Sigv4 (Sigv4a) signing
algorithm.
            val sigV4AScheme = SigV4AsymmetricAuthScheme(CrtAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4AScheme)
            }
            return s3
        }
```

Después de configurar el cliente S3, las operaciones que S3 admite para los puntos de acceso multirregionales funcionan de la misma manera. La única diferencia es que el parámetro del bucket debe ser el del punto ARN de acceso multirregional. Puede obtenerlo ARN desde la consola Amazon S3 o mediante programación, como se muestra anteriormente en la `createMrap` función que devuelve un. ARN

El siguiente ejemplo de código muestra lo que se ARN utiliza en una `GetObject` operación.

```
    suspend fun getObjectFromMrap(
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
            key = keyName
        }

        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
        return stringObj
    }
```

Trabaje con DynamoDB mediante AWS SDK para Kotlin

Utilice puntos de conexión basados AWS en cuentas

DynamoDB AWS ofrece puntos de conexión basados en cuentas que pueden mejorar el rendimiento mediante el uso de AWS su ID de cuenta para agilizar el enrutamiento de solicitudes.

Para aprovechar esta función, debe utilizar la versión 1.3.37 o superior de. AWS SDK para Kotlin Puede encontrar la última versión de la SDK lista en el repositorio central de Maven. Cuando SDK se activa una versión compatible de, utiliza automáticamente los nuevos puntos finales.

Si quieres excluirte del enrutamiento basado en cuentas, tienes cuatro opciones:

- Configure un cliente de servicio de DynamoDB con `AccountIdEndpointMode` el ajuste en. `DISABLED`
- Establezca una variable de entorno.
- Defina una propiedad JVM del sistema.
- Actualice la AWS configuración del archivo de configuración compartido.

El siguiente fragmento es un ejemplo de cómo deshabilitar el enrutamiento basado en cuentas mediante la configuración de un cliente de servicio de DynamoDB:

```
DynamoDbClient.fromEnvironment {
    accountIdEndpointMode = AccountIdEndpointMode.DISABLED // The default value is
 PREFERRED.
}
```

La guía de referencia AWS SDKs y herramientas proporciona más información sobre las tres últimas opciones de configuración.

Asigne clases a elementos de DynamoDB mediante DynamoDB Mapper (versión preliminar para desarrolladores)

⚠ DynamoDB Mapper es una versión preliminar para desarrolladores. No incluye todas las funciones y está sujeta a cambios.

DynamoDB Mapper es una biblioteca de alto nivel que ofrece mecanismos para asignar clases de Kotlin a tablas e índices de DynamoDB, de forma similar al cliente mejorado de DynamoDB o al modelo AWS SDK for Java de persistencia de objetos. AWS SDK for .NET

Defina esquemas que describen su objeto de datos y cómo convertirlos en elementos de DynamoDB. Tras definir el esquema, DynamoDB Mapper proporciona una interfaz intuitiva para utilizar los objetos en las operaciones de creación, lectura, actualización o eliminación CRUD () en las tablas e índices.

Temas

Comience a usar DynamoDB Mapper

> ⚠ DynamoDB Mapper es una versión preliminar para desarrolladores. No incluye todas las funciones y está sujeta a cambios.

En el siguiente tutorial se presentan los componentes básicos de DynamoDB Mapper y se muestra cómo utilizarlos en el código.

agregar dependencias de API

Para empezar a trabajar con DynamoDB Mapper en su proyecto de Gradle, añada un complemento y dos dependencias al archivo. `build.gradle.kts`

(Puedes navegar hasta el *`X.Y.Z`* enlace para ver la última versión disponible).

```
// build.gradle.kts
val sdkVersion: String = X.Y.Z

plugins {
```

```
    id("aws.sdk.kotlin.hll.dynamodbmapper.schema.generator") version "$sdkVersion-
beta" // For the Developer Preview, use the beta version of the latest SDK.
}

dependencies {
    implementation("aws.sdk.kotlin:dynamodb-mapper:$sdkVersion-beta")
    implementation("aws.sdk.kotlin:dynamodb-mapper-annotations:$sdkVersion-beta")
}
```

*<Version>*Sustitúyala por la última versión deSDK. Para encontrar la versión más reciente deSDK, consulte la última versión en GitHub.

> **Note**
>
> Algunas de estas dependencias son opcionales si planea definir los esquemas manualmente. Consulte the section called “Defina esquemas manualmente” para obtener más información y el conjunto reducido de dependencias.

Cree y utilice un mapeador

DynamoDB Mapper utiliza el cliente DynamoDB para AWS SDK para Kotlin interactuar con DynamoDB. Debe proporcionar una instancia completamente configurada al crear una `DynamoDbClient`instancia de mapper, como se muestra en el siguiente fragmento de código:

```
import aws.sdk.kotlin.hll.dynamodbmapper.DynamoDbMapper
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient

val client = DynamoDbClient.fromEnvironment()
val mapper = DynamoDbMapper(client)
```

Una vez que hayas creado la instancia del mapeador, puedes usarla para obtener la instancia de la tabla, como se muestra a continuación:

```
val carsTable = mapper.getTable("cars", CarSchema)
```

El código anterior obtiene una referencia a una tabla DynamoDB nombrada `cars` con un esquema definido por `CarSchema` (analizamos los esquemas a continuación). Después de crear una instancia de tabla, puede realizar operaciones con ella. En el siguiente fragmento de código se muestran dos ejemplos de operaciones con la `cars` tabla:

```
carsTable.putItem {
    item = Car(make = "Ford", model = "Model T", ...)
}

carsTable
   .queryPaginated {
        keyCondition = KeyFilter(partitionKey = "Peugeot")
   }
   .items()
   .collect { car -> println(car) }
```

El código anterior crea un elemento nuevo en la `cars` tabla. El código crea una `Car` instancia en línea utilizando la `Car` clase, cuya definición se muestra a continuación. A continuación, el código consulta la `cars` tabla en busca de elementos cuya clave de partición sea la clave `Peugeot` y los imprime. Las operaciones se describen con más detalle a continuación.

Defina un esquema con anotaciones de clase

Para una variedad de clases de Kotlin, SDK pueden generar esquemas automáticamente en el momento de la compilación mediante el complemento DynamoDB Mapper Schema Generator para Gradle. Cuando utilizas el generador de esquemas, SDK inspecciona tus clases para deducir el esquema, lo que reduce parte del trabajo repetitivo que supone definir esquemas manualmente. Puede personalizar el esquema que se genera mediante anotaciones y configuraciones adicionales.

Para generar un esquema a partir de anotaciones, primero anota tus clases con @`DynamoDbItem` y cualquier clave con y. @`DynamoDbPartitionKey` @`DynamoDbSortKey` El siguiente código muestra la clase `Car` anotada:

```
// The annotations used in the Car class are used by the plugin to generate a schema.
@DynamoDbItem
data class Car(
    @DynamoDbPartitionKey
    val make: String

    @DynamoDbSortKey
    val model: String

    val initialYear: Int
)
```

Después de la construcción, puede consultar la generada `CarSchema` automáticamente. Puedes usar la referencia en el `getTable` método del mapeador para obtener una instancia de tabla, como se muestra a continuación:

```
import aws.sdk.kotlin.hll.dynamodbmapper.generatedschemas.CarSchema

// `CarSchema` is generated at build time.
val carsTable = mapper.getTable("cars", CarSchema)
```

Como alternativa, puedes obtener la instancia de la tabla utilizando un método de extensión `DynamoDbMapper` que se genera automáticamente en el momento de la compilación. Al usar este enfoque, no necesitas referirte al esquema por su nombre. Como se muestra a continuación, el método de `getCarsTable` extensión generado automáticamente devuelve una referencia a la instancia de la tabla:

```
val carsTable = mapper.getCarsTable("cars")
```

Consulte the section called “Genera un esquema” para obtener más detalles y ejemplos.

Invoca operaciones

DynamoDB Mapper admite un subconjunto de las operaciones disponibles en. SDK `DynamoDbClient` Las operaciones de Mapper reciben el mismo nombre que sus homólogas en el cliente. SDK Muchos miembros de la solicitud/respuesta del mapeador son los mismos que sus homólogos SDK clientes, aunque a algunos se les ha cambiado el nombre, se les ha vuelto a escribir o se han eliminado por completo.

Para invocar una operación en una instancia de tabla, utilice una sintaxis como la que se muestra a continuaciónDSL:

```
import aws.sdk.kotlin.hll.dynamodbmapper.operations.putItem
import aws.sdk.kotlin.services.dynamodb.model.ReturnConsumedCapacity

val putResponse = carsTable.putItem {
    item = Car(make = "Ford", model = "Model T", ...)
    returnConsumedCapacity = ReturnConsumedCapacity.Total
}

println(putResponse.consumedCapacity)
```

También puedes invocar una operación mediante un objeto de solicitud explícito:

```
import aws.sdk.kotlin.hll.dynamodbmapper.operations.PutItemRequest
import aws.sdk.kotlin.services.dynamodb.model.ReturnConsumedCapacity

val putRequest = PutItemRequest<Car> {
    item = Car(make = "Ford", model = "Model T", ...)
    returnConsumedCapacity = ReturnConsumedCapacity.Total
}

val putResponse = carsTable.putItem(putRequest)
println(putResponse.consumedCapacity)
```

Los dos ejemplos de código anteriores son equivalentes.

Trabaja con respuestas paginadas

Algunas operaciones, como `query` y `scan` pueden devolver, recopilaciones de datos que pueden ser demasiado grandes como para devolverlas en una sola respuesta. Para garantizar que se procesen todos los objetos, DynamoDB Mapper proporciona métodos de paginación, que no llaman a DynamoDB inmediatamente, sino que devuelven `Flow` un tipo de respuesta a una operación, como se muestra a continuación: `Flow<ScanResponse<Car>>`

```
import aws.sdk.kotlin.hll.dynamodbmapper.operations.scanPaginated

val scanResponseFlow = carsTable.scanPaginated { }

scanResponseFlow.collect { response ->
    val items = response.items.orEmpty()
    println("Found page with ${items.size} items:")

    items.forEach { car -> println(car) }
}
```

A menudo, un flujo de objetos es más útil para la lógica empresarial que un flujo de respuestas que contenga objetos. El mapeador proporciona un método de extensión de las respuestas paginadas para acceder al flujo de objetos. Por ejemplo, el código siguiente devuelve un `Flow<Car>` en lugar de un, `Flow<ScanResponse<Car>>` como se muestra anteriormente:

```
import aws.sdk.kotlin.hll.dynamodbmapper.operations.items
import aws.sdk.kotlin.hll.dynamodbmapper.operations.scanPaginated
```

```
val carFlow = carsTable
    .scanPaginated { }
    .items()

carFlow.collect { car -> println(car) }
```

Configuración de DynamoDB Mapper

⚠ DynamoDB Mapper es una versión preliminar para desarrolladores. No incluye todas las funciones y está sujeta a cambios.

DynamoDB Mapper ofrece opciones de configuración que puede usar para personalizar el comportamiento de la biblioteca para adaptarlo a su aplicación.

Utilice interceptores

La biblioteca DynamoDB Mapper define enlaces que puede utilizar en las etapas críticas de la canalización de solicitudes del mapeador. Puede implementar la `Interceptor` interfaz para implementar enlaces con el fin de observar o modificar el proceso del mapeador.

Puede registrar uno o más interceptores en un único DynamoDB Mapper como opción de configuración. Consulte el ejemplo que aparece al final de esta sección para ver cómo registrar un interceptor.

Comprenda el proceso de solicitudes

La canalización de solicitudes del mapeador consta de los siguientes 5 pasos:

1. Inicialización: configure la operación y recopile el contexto inicial.
2. Serialización: convierte los objetos de solicitud de alto nivel en objetos de solicitud de bajo nivel. Este paso convierte los objetos de Kotlin de alto nivel en elementos de DynamoDB compuestos por nombres y valores de atributos.
3. Invocación de bajo nivel: ejecute una solicitud en el cliente DynamoDB subyacente.
4. Deserialización: convierte los objetos de respuesta de bajo nivel en objetos de respuesta de alto nivel. Este paso incluye la conversión de los elementos de DynamoDB que constan de nombres y valores de atributos en objetos de Kotlin de alto nivel.

5. Finalización: finalice la respuesta de alto nivel para devolverla a la persona que llamó. Si se produjo una excepción durante la ejecución de la canalización, este paso finaliza la excepción que se envía a la persona que llama.

Enlaces

Los ganchos son métodos de intercepción que el mapeador invoca antes o después de pasos específicos de la canalización. Existen dos variantes de ganchos: los de solo lectura y los de modificación (o lectura-escritura). Por ejemplo, `readBeforeInvocation` es un enlace de solo lectura que el mapeador ejecuta en la fase anterior al paso de invocación de bajo nivel.

Ganchos de solo lectura

El mapeador invoca los enlaces de solo lectura antes y después de cada paso del proceso (excepto antes del paso de inicialización y después del paso de finalización). Los capós de solo lectura ofrecen una vista de solo lectura de una operación de alto nivel en curso. Proporcionan un mecanismo para examinar el estado de una operación para, por ejemplo, registrar, depurar o recopilar métricas. Cada enlace de solo lectura recibe un argumento de contexto y lo devuelve. `Unit`

El mapeador detecta cualquier excepción que se produzca durante un enlace de solo lectura y la añade al contexto. A continuación, pasa el contexto, con la excepción de los siguientes ganchos interceptores que se encuentren en la misma fase. El mapeador envía cualquier excepción a la persona que llama solo después de llamar al enlace de solo lectura del último interceptor para la misma fase. Por ejemplo, si un mapeador está configurado con dos interceptores `A` y su `readAfterSerialization` gancho genera una excepción`B`, el mapeador agrega la excepción al contexto pasado al gancho. `A` `B` `readAfterSerialization` Una vez completado `B` `readAfterSerialization` el enlace, el mapeador devuelve la excepción a la persona que llama.

Modifica los ganchos

El mapeador invoca los ganchos de modificación antes de cada paso del proceso (excepto antes de la inicialización). Los ganchos de modificación ofrecen la posibilidad de ver y modificar una operación de alto nivel en curso. Se pueden usar para personalizar el comportamiento y los datos de una forma que no lo hacen la configuración del mapeador y los esquemas de elementos. Cada enlace de modificación recibe un argumento de contexto y, como resultado, devuelve algún subconjunto de ese contexto, ya sea modificado por el enlace o transferido desde el contexto de entrada.

Si el mapeador detecta alguna excepción mientras ejecuta un gancho de modificación, no ejecuta ningún gancho de modificación de ningún otro interceptor en la misma fase. El mapeador

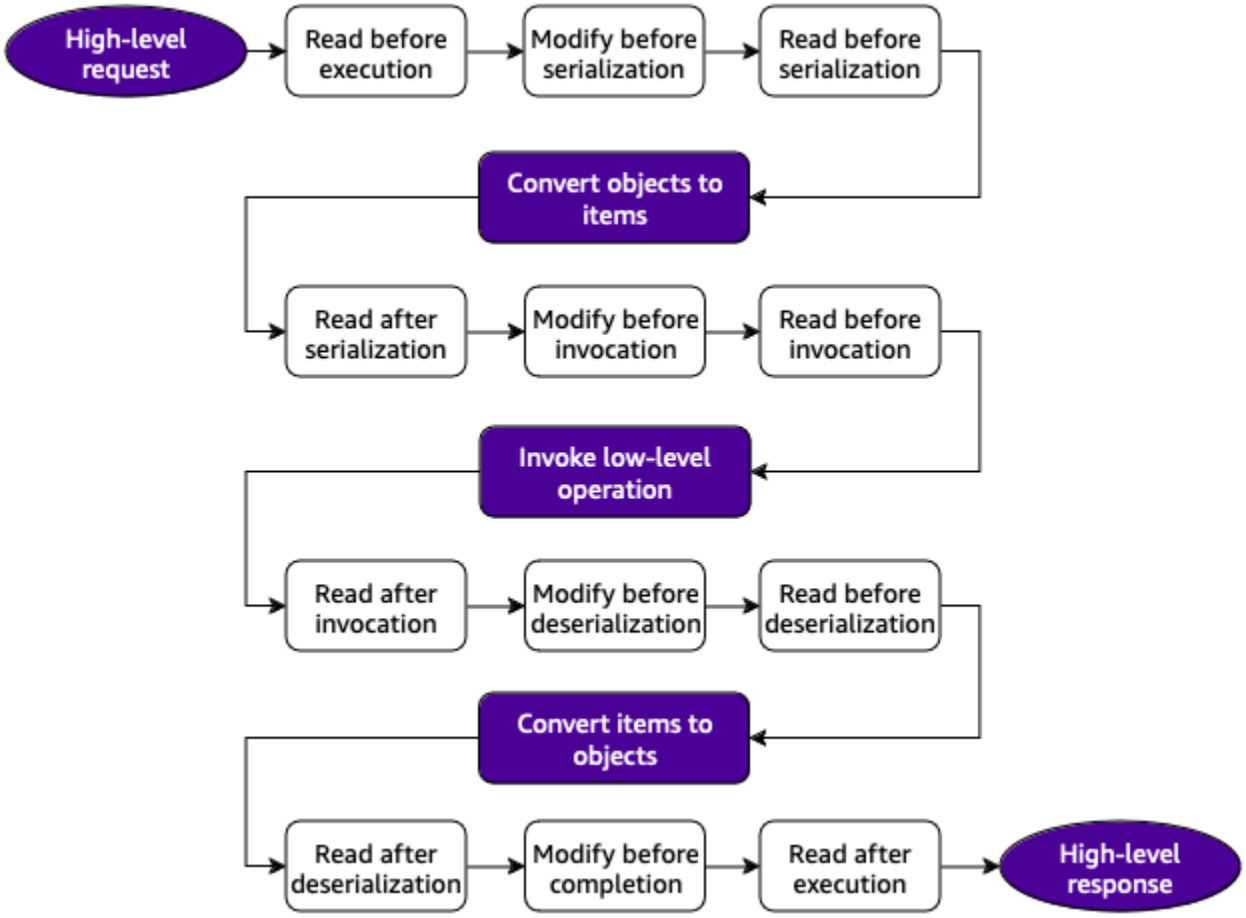
añade la excepción al contexto y la pasa al siguiente enlace de solo lectura. El mapeador envía cualquier excepción a la persona que llama solo después de llamar al enlace de solo lectura del último interceptor para la misma fase. Por ejemplo, si un mapeador está configurado con dos interceptores `A` y su `modifyBeforeSerialization` gancho genera una excepción`B`, `B` no se invocará ese gancho. `A` `modifyBeforeSerialization` Se ejecutarán `A` los interceptores y el `readAfterSerialization` gancho `B'` S, tras lo cual se devolverá la excepción al autor de la llamada.

Orden de ejecución

El orden en que se definen los interceptores en la configuración de un mapeador determina el orden en que el mapeador llama a los ganchos:

- Para las fases anteriores al paso de invocación de bajo nivel, ejecuta los ganchos en el mismo orden en que se agregaron a la configuración.
- Para las fases posteriores al paso de invocación de bajo nivel, ejecuta los ganchos en el orden inverso al orden en que se agregaron a la configuración.

El siguiente diagrama muestra el orden de ejecución de los métodos de enlace:



Descripción textual del orden de ejecución de los métodos de gancho

Un mapeador ejecuta los ganchos de un interceptor en el siguiente orden:

1. DynamoDB Mapper invoca una solicitud de alto nivel
2. Lea antes de la ejecución
3. Modificar antes de la serialización
4. Lea antes de la serialización
5. DynamoDB Mapper convierte objetos en elementos
6. Lea después de la serialización
7. Modificar antes de la invocación
8. Lea antes de la invocación
9. DynamoDB Mapper invoca la operación de bajo nivel
10. Lea después de la invocación
11. Modificar antes de la deserialización

12 Lea antes de la deserialización

13 DynamoDB Mapper convierte elementos en objetos

14 Lea después de la deserialización

15 Modificar antes de finalizar

16 Lea después de la ejecución

17 DynamoDB Mapper devuelve una respuesta de alto nivel

Configuración de ejemplo

El siguiente ejemplo muestra cómo configurar un interceptor en una instancia: `DynamoDbMapper`

```
import aws.sdk.kotlin.hll.dynamodbmapper.DynamoDbMapper
import aws.sdk.kotlin.hll.dynamodbmapper.operations.ScanRequest
import aws.sdk.kotlin.hll.dynamodbmapper.operations.ScanResponse
import aws.sdk.kotlin.hll.dynamodbmapper.pipeline.Interceptor
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import aws.sdk.kotlin.services.dynamodb.model.ScanRequest as LowLevelScanRequest
import aws.sdk.kotlin.services.dynamodb.model.ScanResponse as LowLevelScanResponse

val printingInterceptor = object : Interceptor<User, ScanRequest<User>,
 LowLevelScanRequest, LowLevelScanResponse, ScanResponse<User>> {
    override fun readBeforeDeserialization(ctx: LResContext<User, ScanRequest<User>,
 LowLevelScanRequest, LowLevelScanResponse>) {
        println("Scan response contains ${ctx.lowLevelResponse.count} items.")
    }
}

val client = DynamoDbClient.fromEnvironment()

val mapper = DynamoDbMapper(client) {
    interceptors += printingInterceptor
}
```

Generar un esquema a partir de anotaciones

⚠ DynamoDB Mapper es una versión preliminar para desarrolladores. No incluye todas las funciones y está sujeta a cambios.

DynamoDB Mapper se basa en esquemas que definen el mapeo entre las clases de Kotlin y los elementos de DynamoDB. Las clases de Kotlin pueden impulsar la creación de esquemas mediante el complemento generador de esquemas de Gradle.

Aplica el complemento

Para empezar a generar código para tus clases, aplica el complemento en el script de compilación de la aplicación y añade una dependencia en el módulo de anotaciones. En el siguiente fragmento de script de Gradle, se muestra la configuración necesaria para la generación de código.

(Puedes navegar hasta el *X.Y.Z* enlace para ver la última versión disponible).

```
// build.gradle.kts
val sdkVersion: String = X.Y.Z

plugins {
    id("aws.sdk.kotlin.hll.dynamodbmapper.schema.generator") version "$sdkVersion-
beta" // For the Developer Preview, use the beta version of the latest SDK.
}

dependencies {
    implementation("aws.sdk.kotlin:dynamodb-mapper:$sdkVersion-beta")
    implementation("aws.sdk.kotlin:dynamodb-mapper-annotations:$sdkVersion-beta")
}
```

Configura el complemento

El complemento ofrece una serie de opciones de configuración que puedes aplicar mediante la extensión del `dynamoDbMapper { ... }` complemento en tu script de compilación:

Opción	Descripción de la opción	Valores
`generateBuilderClasses`	Controla si DSL se generarán clases de creación de estilos para las clases anotadas con `@DynamoDbItem`	`WHEN_REQUIRED` (predeterminado): las clases de construcción no se generarán para las clases que consten únicamente de miembros mutables públicos y que tengan un constructor sin arg

Opción	Descripción de la opción	Valores
		`ALWAYS`: Las clases de Builder siempre se generarán
`visibility`	Controla la visibilidad de las clases generadas	`PUBLIC` (predeterminado) `INTERNAL`
`destinationPackage`	Especifica el nombre del paquete de las clases generadas	`RELATIVE`(predeterminado): las clases de esquema se generarán en un subpaquete relativo a la clase anotada. De forma predeterminada, se asigna un nombre al subpaquete`dynamodbmapper.generatedschemas` , que se puede configurar pasando un parámetro de cadena `ABSOLUTE`: Las clases de esquema se generarán en un paquete absoluto relativo a la raíz de la aplicación. De forma predeterminada, el paquete recibe un nombre`aws.sdk.kotlin.hll.dynamodbmapper.generatedschemas` , que se puede configurar pasando un parámetro de cadena.
`generateGetTableExtension`	Controla si se generará un método de `DynamoDbMapper.get${CLASS_NAME}Table` extensión	`true` (predeterminado) `false`

Example Ejemplo de configuración de un complemento de generación de código

El siguiente ejemplo configura el paquete de destino y la visibilidad del esquema generado:

```
// build.gradle.kts

import aws.sdk.kotlin.hll.dynamodbmapper.codegen.annotations.DestinationPackage
import aws.sdk.kotlin.hll.dynamodbmapper.codegen.annotations.Visibility
import aws.smithy.kotlin.runtime.ExperimentalApi

@OptIn(ExperimentalApi::class)
dynamoDbMapper {
    destinationPackage = DestinationPackage.RELATIVE("my.configured.package")
    visibility = Visibility.INTERNAL
}
```

Anota las clases

El generador de esquemas busca anotaciones de clases para determinar para qué clases generar esquemas. Para optar por la generación de esquemas, anota tus clases con ellas. @DynamoDbItem También debes anotar con la anotación una propiedad de clase que sirva como clave de partición del elemento. @DynamoDbPartitionKey

La siguiente definición de clase muestra las anotaciones mínimas necesarias para la generación del esquema:

Example

```
@DynamoDbItem
data class Employee(
    @DynamoDbPartitionKey
    val id: Int,

    val name: String,
    val role: String,
)
```

Anotaciones de clase

Las siguientes anotaciones se aplican a las clases para controlar la generación del esquema:

- `@DynamoDbItem`: Especifica que esta clase/interfaz describe un tipo de elemento de una tabla. Todas las propiedades públicas de este tipo se asignarán a atributos a menos que se ignoren de forma explícita. Cuando esté presente, se generará un esquema para esta clase.
  - `converterName`: Un parámetro opcional que indica que se debe usar un esquema personalizado en lugar del creado por el complemento generador de esquemas. Este es el nombre completo de la `ItemConverter` clase personalizada. the section called “Defina un conversor de artículos personalizado”En la sección se muestra un ejemplo de creación y uso de un esquema personalizado.

Anotaciones de propiedades

Puede aplicar las siguientes anotaciones a las propiedades de la clase para controlar la generación del esquema:

- @`DynamoDbPartitionKey`: Especifica la clave de partición del elemento.
- @`DynamoDbSortKey`: especifica una clave de clasificación opcional para el elemento.
- @`DynamoDbIgnore`: especifica que DynamoDB Mapper no debe convertir esta propiedad de clase hacia/desde un atributo de elemento.
- @`DynamoDbAttribute`: especifica un nombre de atributo personalizado opcional para esta propiedad de clase.

Defina un conversor de artículos personalizado

En algunos casos, es posible que desees definir un conversor de artículos personalizado para tu clase. Una razón para ello sería si tu clase usa un tipo que no es compatible con el complemento generador de esquemas. Usamos la siguiente versión de la `Employee` clase como ejemplo:

```
import kotlin.uuid.Uuid

@DynamoDbItem
data class Employee(
    @DynamoDbPartitionKey
    var id: Int,

    var name: String,
    var role: String,
    var workstationId: Uuid
)
```

La `Employee` clase ahora usa un `kotlin.uuid.Uuid` tipo, que actualmente no es compatible con el generador de esquemas. La generación del esquema falla y se produce un error:`Unsupported attribute type TypeRef(pkg=kotlin.uuid, shortName=Uuid, genericArgs=[], nullable=false)`. Este error indica que el complemento no puede generar un convertidor de elementos para esta clase. Por lo tanto, necesitamos escribir el nuestro.

Para ello, implementamos una `ItemConverter` para la clase y, a continuación, modificamos la anotación de la `@DynamoDbItem` clase especificando el nombre completo del nuevo conversor de elementos.

En primer lugar, implementamos a `ValueConverter` para la `kotlin.uuid.Uuid` clase:

```
import aws.sdk.kotlin.hll.dynamodbmapper.values.ValueConverter
import aws.sdk.kotlin.services.dynamodb.model.AttributeValue
import kotlin.uuid.Uuid

public val UuidValueConverter = object : ValueConverter<Uuid> {
    override fun convertFrom(to: AttributeValue): Uuid =
        Uuid.parseHex(to.asS())

    override fun convertTo(from: Uuid): AttributeValue =
        AttributeValue.S(from.toHexString())
}
```

Luego, implementamos una `ItemConverter` para nuestra `Employee` clase. `ItemConverter`Utiliza este nuevo conversor de valores en el descriptor de atributos de "workstationId«:

```
import aws.sdk.kotlin.hll.dynamodbmapper.items.AttributeDescriptor
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.items.SimpleItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.values.scalars.IntConverter
import aws.sdk.kotlin.hll.dynamodbmapper.values.scalars.StringConverter

public object MyEmployeeConverter : ItemConverter<Employee> by SimpleItemConverter(
    builderFactory = { Employee() },
    build = { this },
    descriptors = arrayOf(
        AttributeDescriptor(
            "id",
            Employee::id,
```

```
            Employee::id::set,
            IntConverter,
        ),
        AttributeDescriptor(
            "name",
            Employee::name,
            Employee::name::set,
            StringConverter,
        ),
        AttributeDescriptor(
            "role",
            Employee::role,
            Employee::role::set,
            StringConverter
        ),
        AttributeDescriptor(
            "workstationId",
            Employee::workstationId,
            Employee::workstationId::set,
            UuidValueConverter
        )
    ),
)
```

Ahora que hemos definido el conversor de artículos, podemos aplicarlo a nuestra clase. Actualizamos la @DynamoDbItem anotación para que haga referencia al conversor de artículos proporcionando el nombre completo de la clase, tal y como se muestra a continuación:

```
import kotlin.uuid.Uuid

@DynamoDbItem("my.custom.item.converter.MyEmployeeConverter")
data class Employee(
    @DynamoDbPartitionKey
    var id: Int,

    var name: String,
    var role: String,
    var workstationId: Uuid
)
```

Por último, podemos empezar a usar la clase con DynamoDB Mapper.

Defina esquemas manualmente

⚠ DynamoDB Mapper es una versión preliminar para desarrolladores. No incluye todas las funciones y está sujeta a cambios.

Defina un esquema en el código

Para obtener el máximo control y personalización, puede definir y personalizar manualmente los esquemas en el código.

Como se muestra en el siguiente fragmento, es necesario incluir menos dependencias en el `build.gradle.kts` archivo en comparación con la creación de esquemas basada en anotaciones.

(Puedes navegar hasta el *X.Y.Z* enlace para ver la última versión disponible).

```
// build.gradle.kts
val sdkVersion: String = X.Y.Z

dependencies {
    implementation("aws.sdk.kotlin:dynamodb-mapper:$sdkVersion-beta") // For the
 Developer Preview, use the beta version of the latest SDK.
}
```

Ten en cuenta que no necesitas el complemento generador de esquemas ni el paquete de anotaciones.

El mapeo entre una clase de Kotlin y un elemento de DynamoDB requiere `ItemSchema<T>` una implementación, T donde es el tipo de la clase de Kotlin. Un esquema consta de los siguientes elementos:

- Un conversor de elementos, que define cómo convertir entre instancias de objetos de Kotlin y elementos de DynamoDB.
- Una especificación de clave de partición, que define el nombre y el tipo del atributo de clave de partición.
- Opcionalmente, una especificación de clave de clasificación, que define el nombre y el tipo del atributo de clave de clasificación.

En el siguiente código, creamos una `CarSchema` instancia manualmente:

```
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemSchema
import aws.sdk.kotlin.hll.dynamodbmapper.model.itemOf

// We define a schema for this data class.
data class Car(val make: String, val model: String, val initialYear: Int)

// First, define an item converter.
val carConverter = object : ItemConverter<Car> {
    override fun convertTo(from: Car, onlyAttributes: Set<String>?): Item  = itemOf(
        "make" to AttributeValue.S(from.make),
        "model" to AttributeValue.S(from.model),
        "initialYear" to AttributeValue.N(from.initialYear.toString()),
    )

    override fun convertFrom(to: Item): Car = Car(
        make = to["make"]?.asSOrNull() ?: error("Invalid attribute `make`"),
        model = to["model"]?.asSOrNull() ?: error("Invalid attribute `model`"),
        initialYear = to["initialYear"]?.asNOrNull()?.toIntOrNull()
            ?: error("Invalid attribute `initialYear`"),
    )
}

// Next, define the specifications for the partition key and sort key.
val makeKey = KeySpec.String("make")
val modelKey = KeySpec.String("model")

// Finally, create the schema from the converter and key specifications.
// Note that the KeySpec for the partition key comes first in the ItemSchema
 constructor.
val CarSchema = ItemSchema(carConverter, makeKey, modelKey)
```

El código anterior crea un convertidor denominado`carConverter`, que se define como una implementación anónima de`ItemConverter<Car>`. El `convertTo` método del convertidor acepta un `Car` argumento y devuelve una `Item` instancia que representa las claves y valores literales de los atributos de los elementos de DynamoDB. El `convertFrom` método del convertidor acepta un `Item` argumento y devuelve una `Car` instancia a partir de los valores de atributo del `Item` argumento.

A continuación, el código crea dos especificaciones clave: una para la clave de partición y otra para la clave de clasificación. Cada tabla o índice de DynamoDB debe tener exactamente una clave de partición y, en consecuencia, también lo debe tener cada definición de esquema de DynamoDB Mapper. Los esquemas también pueden tener una clave de clasificación.

En la última declaración, el código crea un esquema para la tabla de `cars` DynamoDB a partir del convertidor y de las especificaciones clave.

El esquema resultante es equivalente al esquema basado en anotaciones que generamos en la sección. the section called “Defina un esquema con anotaciones de clase” Como referencia, la siguiente es la clase anotada que utilizamos:

Clase de vehículo con anotaciones en DynamoDB Mapper

```
@DynamoDbItem
data class Car(
    @DynamoDbPartitionKey
    val make: String

    @DynamoDbSortKey
    val model: String

    val initialYear: Int
)
```

Además de implementar las suyas propias`ItemConverter`, DynamoDB Mapper incluye varias implementaciones útiles, como:

- `SimpleItemConverter`: proporciona una lógica de conversión sencilla mediante el uso de un generador de descriptores de atributos y clases. Consulte el the section called “Defina un conversor de artículos personalizado” ejemplo de cómo puede utilizar esta implementación.
- `HeterogeneousItemConverter`: proporciona una lógica de conversión de tipos polimórficos mediante el uso de un atributo discriminador y `ItemConverter` instancias delegadas para los subtipos.
- `DocumentConverter`: proporciona lógica de conversión para datos no estructurados en objetos. `Document`

Utilice índices secundarios con DynamoDB Mapper

⚠ DynamoDB Mapper es una versión preliminar para desarrolladores. No incluye todas las funciones y está sujeta a cambios.

Defina un esquema para un índice secundario

Las tablas de DynamoDB admiten índices secundarios que proporcionan acceso a los datos mediante claves distintas de las definidas en la propia tabla base. Al igual que con las tablas base, DynamoDB Mapper interactúa con los índices mediante el tipo. `ItemSchema`

No es necesario que los índices secundarios de DynamoDB contengan todos los atributos de la tabla base. En consecuencia, la clase de Kotlin que se asigna a un índice puede diferir de la clase de Kotlin que se asigna a la tabla base de ese índice. En ese caso, se debe declarar un esquema independiente para la clase de índice.

El código siguiente crea manualmente un esquema de índice para la tabla de `cars` DynamoDB.

```
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemSchema
import aws.sdk.kotlin.hll.dynamodbmapper.model.itemOf

// This is a data class for modelling the index of the Car table. Note
// that it contains a subset of the fields from the Car class and also
// uses different names for them.
data class Model(val name: String, val manufacturer: String)

// We define an item converter.
val modelConverter = object : ItemConverter<Model> {
    override fun convertTo(from: Model, onlyAttributes: Set<String>?): Item  = itemOf(
        "model" to AttributeValue.S(from.name),
        "make" to AttributeValue.S(from.manufacturer),
    )

    override fun convertFrom(to: Item): Model = Model(
        name = to["model"]?.asSOrNull() ?: error("Invalid attribute `model`"),
        manufacturer = to["make"]?.asSOrNull() ?: error("Invalid attribute `make`"),
    )
}
val modelKey = KeySpec.String("model")
val makeKey = KeySpec.String("make")

val modelSchema = ItemSchema(modelConverter, modelKey, makeKey) // The partition key
 specification is the second parameter.

/* Note that `Model` index's partition key is `model` and its sort key is `make`,
   whereas the `Car` base table uses `make` as the partition key and `model` as the
 sort key:
```

```
        @DynamoDbItem
        data class Car(
            @DynamoDbPartitionKey
            val make: String

            @DynamoDbSortKey
            val model: String

            val initialYear: Int
        )
*/
```

Ahora podemos usar `Model` instancias en las operaciones.

Utilice índices secundarios en las operaciones

DynamoDB Mapper admite un subconjunto de operaciones en índices, a saber, y. `queryPaginated` `scanPaginated` Para invocar estas operaciones en un índice, primero debe obtener una referencia a un índice desde el objeto de la tabla. En el siguiente ejemplo, utilizamos lo `modelSchema` que creamos anteriormente para el `cars-by-model` índice (la creación no se muestra aquí):

```
val table = mapper.getTable("cars", CarSchema)
val index = table.getIndex("cars-by-model", modelSchema)

val modelFlow = index
    .scanPaginated { }
    .items()

modelFlow.collect { model -> println(model) }
```

Usa expresiones

> ⚠ DynamoDB Mapper es una versión preliminar para desarrolladores. No incluye todas las funciones y está sujeta a cambios.

Algunas operaciones de DynamoDB aceptan expresiones que se pueden usar para especificar restricciones o condiciones. DynamoDB Mapper proporciona un Kotlin idiomático para crear

expresionesDSL. DSLEsto aporta una mayor estructura y legibilidad al código y también facilita la escritura de expresiones.

En esta sección se describe la DSL sintaxis y se proporcionan varios ejemplos.

Usa expresiones en las operaciones

Usas expresiones en operaciones como`scan`, por ejemplo, en las que filtran los elementos devueltos en función de los criterios que tú definas. Para usar expresiones con DynamoDB Mapper, añada el componente de expresión en la solicitud de operación.

En el siguiente fragmento se muestra un ejemplo de una expresión de filtro que se utiliza en una operación. `scan` Utiliza un argumento lambda para describir los criterios de filtro que limitan los elementos que se van a devolver a aquellos con un valor de `year` atributo de 2001:

```
val table = // A table instance.

table.scanPaginated {
    filter {
        attr("year") eq 2001
    }
}
```

El siguiente ejemplo muestra una `query` operación que admite expresiones en dos lugares: filtrado por clave de clasificación y filtrado sin clave:

```
table.queryPaginated {
    keyCondition = KeyFilter(partitionKey = 1000) { sortKey startsWith "M" }
    filter {
        attr("year") eq 2001
    }
}
```

El código anterior filtra los resultados para que cumplan los tres criterios:

- El valor del atributo clave de partición es 1000 - AND -
- El valor del atributo de la clave de clasificación comienza con la letra M - AND -
- el valor del atributo del año es 2001

Componentes de DSL

La DSL sintaxis expone varios tipos de componentes (que se describen a continuación) que se utilizan para crear expresiones.

Atributos

La mayoría de las condiciones hacen referencia a los atributos, que se identifican por su clave o ruta de documento. Con elDSK, puede crear todas las referencias de atributos mediante la `attr` función y, si lo desea, realizar modificaciones adicionales.

El código siguiente muestra un rango de ejemplos de referencias a atributos, desde simples hasta complejas, como la desreferenciación de listas por índice y la desreferenciación de mapas por clave:

```
attr("foo")           // Refers to the value of top-level attribute `foo`.

attr("foo")[3]        // Refers to the value at index 3 in the list value of
                      // attribute `foo`.

attr("foo")[3]["bar"] // Refers to the value of key `bar` in the map value at
                      // index 3 of the list value of attribute `foo`.
```

Igualidades y desigualdades

Puede comparar los valores de los atributos de una expresión mediante igualdades y desigualdades. Puede comparar los valores de los atributos con valores literales u otros valores de atributos. Las funciones que se utilizan para especificar las condiciones son:

- `eq`: es igual a (equivalente a==)
- `neq`: no es igual a (equivalente a!=)
- `gt`: es mayor que (equivalente a>)
- `gte`: es mayor o igual a (equivalente a>=)
- `lt`: es menor que (equivalente a<)
- `lte`: es menor o igual a (equivalente a<=)

La función de comparación con los argumentos se combina mediante la notación infija, como se muestra en los siguientes ejemplos:

```
attr("foo") eq 42           // Uses a literal. Specifies that the attribute value `foo`
 must be
                            // equal to 42.

attr("bar") gte attr("baz") // Uses another attribute value. Specifies that the
 attribute
                            // value `bar` must be greater than or equal to the
                            // attribute value of `baz`.
```

Rangos y conjuntos

Además de los valores individuales, puede comparar los valores de los atributos con varios valores de los rangos o conjuntos. La `isIn` función infijo se utiliza para realizar la comparación, tal y como se muestra en los siguientes ejemplos:

```
attr("foo") isIn 0..99  // Specifies that the attribute value `foo` must be
                        // in the range of `0` to `99` (inclusive).

attr("foo") isIn setOf( // Specifies that the attribute value `foo` must be
    "apple",            // one of `apple`, `banana`, or `cherry`.
    "banana",
    "cherry",
)
```

La `isIn` función proporciona sobrecargas para las colecciones (por ejemplo`Set<String>`) y para los límites que se pueden expresar como Kotlin `ClosedRange<T>` (por ejemplo). `IntRange` Para los límites que no puedes expresar como a `ClosedRange<T>` (como matrices de bytes u otras referencias de atributos), puedes usar la función: `isBetween`

```
val lowerBytes = byteArrayOf(0x48, 0x65, 0x6c)  // Specifies that the attribute value
val upperBytes = byteArrayOf(0x6c, 0x6f, 0x21)  // `foo` is between the values
attr("foo").isBetween(lowerBytes, upperBytes)   // `0x48656c` and `0x6c6f21`

attr("foo").isBetween(attr("bar"), attr("baz")) // Specifies that the attribute value
                                                // `foo` is between the values of
                                                // attributes `bar` and `baz`.
```

Lógica booleana

Puede combinar condiciones individuales o modificarlas mediante la lógica booleana mediante las siguientes funciones:

- `and`: todas las condiciones deben ser verdaderas (equivalentes a) `&&`
- `or`: al menos una condición debe ser verdadera (equivalente a `||`)
- `not`: la condición dada debe ser falsa (equivalente a `!`)

Los siguientes ejemplos muestran cada función:

```
and(                            // Both conditions must be met:
    attr("foo") eq "banana",    // * attribute value `foo` must equal `banana`
    attr("bar") isIn 0..99,     // * attribute value `bar` must be between
)                               //   0 and 99 (inclusive)

or(                             // At least one condition must be met:
    attr("foo") eq "cherry",    // * attribute value `foo` must equal `cherry`
    attr("bar") isIn 100..199, // * attribute value `bar` must be between
)                               //   100 and 199 (inclusive)

not(                            // The attribute value `foo` must *not* be
    attr("baz") isIn setOf(     // one of `apple`, `banana`, or `cherry`.
        "apple",                // Stated another way, the attribute value
        "banana",               // must be *anything except* `apple`, `banana`,
        "cherry",               // or `cherry`--including potentially a
    ),                          // non-string value or no value at all.
)
```

También puede combinar condiciones booleanas con funciones booleanas para crear una lógica anidada, como se muestra en la siguiente expresión:

```
or(
    and(
        attr("foo") eq 123,
        attr("bar") eq "abc",
    ),
    and(
        attr("foo") eq 234,
        attr("bar") eq "bcd",
    ),
)
```

La expresión anterior filtra los resultados por aquellos que cumplen alguna de estas condiciones:

- Se cumplen las dos condiciones siguientes:

  - `foo`el valor del atributo es 123 - AND -
  - `bar`el valor del atributo es «abc»
- Ambas condiciones son verdaderas:
  - `foo`el valor del atributo es 234 - AND -
  - `bar`el valor del atributo es «bcd»

Esto equivale a la siguiente expresión booleana de Kotlin:

```
(foo == 123 && bar == "abc") || (foo == 234 && bar == "bcd")
```

Funciones y propiedades

Las siguientes funciones y propiedades proporcionan capacidades de expresión adicionales:

- `contains`: comprueba si el valor de un atributo de cadena o lista contiene un valor determinado
- `exists`: comprueba si un atributo está definido y contiene algún valor (incluido) `null`
- `notExists`: comprueba si un atributo no está definido
- `isOfType`: comprueba si el valor de un atributo es de un tipo determinado, como cadena, número, booleano, etc.
- `size`: obtiene el tamaño de un atributo, como el número de elementos de una colección o la longitud de una cadena
- `startsWith`: comprueba si el valor de un atributo de cadena comienza con una subcadena determinada

Los siguientes ejemplos muestran el uso de funciones y propiedades adicionales que se pueden usar en expresiones:

```
attr("foo") contains "apple" // Specifies that the attribute value `foo` must be
                             // a list that contains an `apple` element or a string
                             // which contains the substring `apple`.

attr("bar").exists()         // Specifies that the `bar` must exist and have a
                             // value (including potentially `null`).

attr("baz").size lt 100      // Specifies that the attribute value `baz` must have
                             // a size of less than 100.
```

```
attr("qux") isOfType AttributeType.String // Specifies that the attribute `qux`
                                          // must have a string value.
```

Ordene los filtros clave

Las expresiones de filtro de las claves de clasificación (como las del `keyCondition` parámetro de la `query` operación) no utilizan valores de atributos con nombre asignado. Para utilizar una clave de ordenación en un filtro, debe utilizar la palabra clave `sortKey` en todas las comparaciones. La `sortKey` palabra clave reemplaza `attr("<sort key name>")` como se muestra en los ejemplos siguientes:

```
sortKey startsWith "abc" // The sort key attribute value must begin with the
                         // substring `abc`.

sortKey isIn 0..99       // The sort key attribute value must be between 0
                         // and 99 (inclusive).
```

Los filtros de claves de clasificación no se pueden combinar con la lógica booleana y solo admiten un subconjunto de las comparaciones descritas anteriormente:

- Igualidades y desigualdades: se admiten todas las comparaciones
- Rangos y conjuntos: se admiten todas las comparaciones
- Lógica booleana: no se admite
- Funciones y propiedades: solo se admite `startsWith`

SDKpara ejemplos de código de Kotlin

Los ejemplos de código de este tema te muestran cómo usar el código AWS SDK para Kotlin con. AWS

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Algunos servicios contienen categorías de ejemplo adicionales que muestran cómo aprovechar las bibliotecas o funciones específicas del servicio.

Servicios

- APIEjemplos de gateway que se utilizan SDK para Kotlin
- Ejemplos de Aurora utilizando SDK para Kotlin
- Ejemplos de Auto Scaling que se utilizan SDK para Kotlin
- Ejemplos de uso de Amazon Bedrock SDK para Kotlin
- CloudWatch ejemplos que utilizan SDK para Kotlin
- CloudWatch Registra ejemplos que se utilizan SDK para Kotlin
- Ejemplos de proveedores de identidad de Amazon Cognito que se utilizan SDK para Kotlin
- Ejemplos de Amazon Comprehend utilizando Kotlin SDK
- Ejemplos de DynamoDB que utilizan para Kotlin SDK
- EC2Ejemplos de Amazon que utilizan SDK Kotlin
- ECREjemplos de Amazon que utilizan SDK Kotlin
- OpenSearch Ejemplos de servicios que se utilizan SDK para Kotlin
- EventBridge ejemplos que utilizan SDK para Kotlin
- AWS Glue ejemplos que utilizan SDK para Kotlin
- IAMejemplos que utilizan SDK para Kotlin

- AWS IoT ejemplos que utilizan SDK para Kotlin
- AWS IoT data ejemplos que utilizan SDK para Kotlin
- Ejemplos de Amazon Keyspaces que se utilizan SDK para Kotlin
- AWS KMS ejemplos que utilizan SDK para Kotlin
- Ejemplos de Lambda que se utilizan SDK para Kotlin
- MediaConvert ejemplos que se utilizan SDK para Kotlin
- Ejemplos de Amazon Pinpoint que utilizan Kotlin SDK
- RDSEjemplos de Amazon que utilizan SDK Kotlin
- Ejemplos RDS de Amazon Data Service que utilizan SDK Kotlin
- Ejemplos de uso SDK de Amazon Redshift para Kotlin
- Ejemplos de Amazon Rekognition que utilizan SDK Kotlin
- Ejemplos de registro de dominios de Route 53 utilizando SDK Kotlin
- Ejemplos de Amazon S3 que utilizan SDK Kotlin
- SageMaker Ejemplos de IA que se utilizan SDK para Kotlin
- Ejemplos de Secrets Manager que se utilizan SDK para Kotlin
- SESEjemplos de Amazon que utilizan SDK Kotlin
- SNSEjemplos de Amazon que utilizan SDK Kotlin
- SQSEjemplos de Amazon que utilizan SDK Kotlin
- Ejemplos de Step Functions que se utilizan SDK para Kotlin
- Support ejemplos que utilizan SDK para Kotlin
- Ejemplos de Amazon Translate que utilizan SDK Kotlin

APIEjemplos de gateway que se utilizan SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con API Gateway.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Escenarios

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDKpara Kotlin

Muestra cómo desarrollar una aplicación de administración 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

- Gateway de API
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Ejemplos de Aurora utilizando SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con Aurora.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Conceptos básicos

Conceptos básicos

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.

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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

This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster 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 the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
```

```
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
```

```
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

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

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

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
```

```
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

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

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

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

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
```

```
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

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

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

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

    var endpoint = ""
```

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

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

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

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
```

```
    var instanceClass = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

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

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

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

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
```

```
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

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

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

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

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

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

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

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

```
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

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

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

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

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

```
suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - Instantánea C reateDBCluster
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - escribeDBClusterParámetros D
  - escribeDBClusterInstantáneas en 3D
  - DescribeDBClusters
  - Versiones en 3D escribeDBEngine
  - DescribeDBInstances

- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Acciones

`CreateDBCluster`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateDBCluster`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- Para API obtener más información, consulta C reateDBCluster en la APIreferencia AWS SDK a Kotlin.

`CreateDBClusterParameterGroup`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateDBClusterParameterGroup`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para API obtener más información, consulta C reateDBCluster ParameterGroup en la APIreferencia AWS SDK a Kotlin.

`CreateDBClusterSnapshot`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateDBClusterSnapshot`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para API obtener más información, consulta C reateDBCluster Snapshot en la sección AWS SDKde API referencia sobre Kotlin.

`CreateDBInstance`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateDBInstance`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para API obtener más información, consulta C reateDBInstance en la APIreferencia AWS SDK a Kotlin.

`DeleteDBCluster`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteDBCluster`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

```
        }

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

- Para API obtener más información, consulte D eleteDBCluster en la APIreferencia AWS SDK de Kotlin.

`DeleteDBClusterParameterGroup`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteDBClusterParameterGroup`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
```

```
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

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

- Para API obtener más información, consulte D eleteDBCluster ParameterGroup en la APIreferencia AWS SDK de Kotlin.

`DeleteDBInstance`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteDBInstance`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para API obtener más información, consulte D eleteDBInstance en la APIreferencia AWS SDK de Kotlin.

`DescribeDBClusterParameterGroups`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeDBClusterParameterGroups`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para API obtener más información, consulte D escribeDBCluster ParameterGroups en la APIreferencia AWS SDK de Kotlin.

`DescribeDBClusterParameters`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeDBClusterParameters`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
```

```
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

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

- Para API obtener más información, consulta escribeDBClusterlos parámetros D en la sección AWS SDKde API referencia sobre Kotlin.

`DescribeDBClusterSnapshots`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeDBClusterSnapshots`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

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

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

- Para API obtener más información, consulta D escribeDBCluster Snapshots en AWS SDK la sección de referencia sobre Kotlin API.

`DescribeDBClusters`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeDBClusters`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

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

- Para API obtener más información, consulte D escribeDBClusters en la APIreferencia AWS SDK de Kotlin.

`DescribeDBEngineVersions`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeDBEngineVersions`.

SDKpara Kotlin

> **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.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

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

```
        }
    }
}
```

- Para API obtener más información, consulta escribeDBEnginelas versiones D en la sección AWS SDKde API referencia sobre Kotlin.

`DescribeDBInstances`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeDBInstances`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

```
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}
```

- Para API obtener más información, consulte D escribeDBInstances en la APIreferencia AWS SDK de Kotlin.

`ModifyDBClusterParameterGroup`

En el siguiente ejemplo de código, se muestra cómo utilizar `ModifyDBClusterParameterGroup`.

SDKpara Kotlin

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

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

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

```
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- Para API obtener más información, consulte M odifyDBCluster ParameterGroup en la APIreferencia AWS SDK de Kotlin.

Escenarios

Crear un rastreador de elementos de trabajo de Aurora Serverless

El siguiente ejemplo de código muestra cómo crear una aplicación web que haga un seguimiento de los elementos de trabajo de una base de datos Amazon Aurora Serverless y utilice Amazon Simple Email Service (AmazonSES) para enviar informes.

SDKpara Kotlin

Muestra cómo crear una aplicación web que rastrea e informa sobre los elementos de trabajo almacenados en una RDS base de datos de Amazon.

Para obtener el código fuente completo y las instrucciones sobre cómo configurar un Spring REST API que consulte los datos de Amazon Aurora Serverless y para que lo utilice una aplicación de React, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Ejemplos de Auto Scaling que se utilizan SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin with Auto Scaling.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Cree un grupo de Amazon EC2 Auto Scaling con una plantilla de lanzamiento y zonas de disponibilidad, y obtenga 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.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
        launchTemplateName - The name of the launch template.
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-linked
 role that the Auto Scaling group uses.
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances in
 the Auto Scaling group can be created.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("**** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
 vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("**** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)
```

```
    println("**** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("**** Update an Auto Scaling group to maximum size of 3")
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("**** Describe all Auto Scaling groups to show the current state of the
 groups")
    describeAutoScalingGroups(groupName)

    println("**** Describe account details")
    describeAccountLimits()

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    println("**** Set desired capacity to 2")
    setDesiredCapacity(groupName)

    println("**** Get the two instance Id values and state")
    getAutoScalingGroups(groupName)

    println("**** List the scaling activities that have occurred for the group")
    describeScalingActivities(groupName)

    println("**** Terminate an instance in the Auto Scaling group")
    terminateInstanceInAutoScalingGroup(instanceId)

    println("**** Stop the metrics collection")
    disableMetricsCollection(groupName)

    println("**** Delete the Auto Scaling group")
    deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
```

```
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }
```

```
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }
```

```
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
```

```
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
```

```
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
    return instanceId
}

suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
 ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
 ${response.numberOfAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
```

```
        println("You successfully deleted $groupName")
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Acciones

`CreateAutoScalingGroup`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateAutoScalingGroup`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
```

```
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}
```

- Para API obtener más información, consulta CreateAutoScalingGroupla AWS SDKAPIreferencia sobre Kotlin.

`DeleteAutoScalingGroup`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteAutoScalingGroup`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- Para API obtener más información, consulta DeleteAutoScalingGroupla AWS SDKAPIreferencia sobre Kotlin.

`DescribeAutoScalingGroups`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeAutoScalingGroups`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
```

```
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}
```

- Para API obtener más información, consulta DescribeAutoScalingGroupsla AWS SDKAPIreferencia sobre Kotlin.

`DescribeAutoScalingInstances`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeAutoScalingInstances`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
```

```
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- Para API obtener más información, consulta DescribeAutoScalingInstancesla AWS SDKAPIreferencia sobre Kotlin.

`DescribeScalingActivities`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeScalingActivities`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}
```

- Para API obtener más información, consulta DescribeScalingActivitiesla AWS SDKAPIreferencia sobre Kotlin.

`DisableMetricsCollection`

En el siguiente ejemplo de código, se muestra cómo utilizar `DisableMetricsCollection`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- Para API obtener más información, consulta DisableMetricsCollectionla AWS SDKAPIreferencia sobre Kotlin.

`EnableMetricsCollection`

En el siguiente ejemplo de código, se muestra cómo utilizar `EnableMetricsCollection`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- Para API obtener más información, consulta EnableMetricsCollectionla AWS SDKAPIreferencia sobre Kotlin.

`SetDesiredCapacity`

En el siguiente ejemplo de código, se muestra cómo utilizar `SetDesiredCapacity`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
```

```
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- Para API obtener más información, consulta SetDesiredCapacityla AWS SDKAPIreferencia sobre Kotlin.

`TerminateInstanceInAutoScalingGroup`

En el siguiente ejemplo de código, se muestra cómo utilizar `TerminateInstanceInAutoScalingGroup`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- Para API obtener más información, consulta TerminateInstanceInAutoScalingGroupla AWS SDKAPIreferencia sobre Kotlin.

`UpdateAutoScalingGroup`

En el siguiente ejemplo de código, se muestra cómo utilizar `UpdateAutoScalingGroup`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
```

```
    }
}
```

- Para API obtener más información, consulta UpdateAutoScalingGroupla AWS SDKAPIreferencia sobre Kotlin.

Ejemplos de uso de Amazon Bedrock SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- Acciones

Acciones

`ListFoundationModels`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListFoundationModels`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumerar los modelos fundacionales de Amazon Bedrock disponibles.

```
suspend fun listFoundationModels(): List<FoundationModelSummary>? {
```

```
    BedrockClient { region = "us-east-1" }.use { bedrockClient ->
        val response =
 bedrockClient.listFoundationModels(ListFoundationModelsRequest {})
        response.modelSummaries?.forEach { model ->
            println("==========================================")
            println(" Model ID: ${model.modelId}")
            println("------------------------------------------")
            println(" Name: ${model.modelName}")
            println(" Provider: ${model.providerName}")
            println(" Input modalities: ${model.inputModalities}")
            println(" Output modalities: ${model.outputModalities}")
            println(" Supported customizations: ${model.customizationsSupported}")
            println(" Supported inference types: ${model.inferenceTypesSupported}")
            println("------------------------------------------\n")
        }
        return response.modelSummaries
    }
}
```

- Para API obtener más información, consulta ListFoundationModelsla AWS SDKAPIreferencia sobre Kotlin.

CloudWatch ejemplos que utilizan SDK para Kotlin

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Kotlin with. CloudWatch

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Introducción

Hola CloudWatch

En los siguientes ejemplos de código se muestra cómo empezar a utilizar CloudWatch.

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <namespace>
        Where:
           namespace - The namespace to filter against (for example, AWS/EC2).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val namespace = args[0]
    listAllMets(namespace)
}

suspend fun listAllMets(namespaceVal: String?) {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
```

```
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listMetricsPaginated(request)
            .transform { it.metrics?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.metricName}")
                println("Namespace is ${obj.namespace}")
            }
    }
}
```

- Para API obtener más información, consulta ListMetricsla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Conceptos básicos
- Acciones

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Enumera los CloudWatch espacios de 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.

SDKpara Kotlin

> **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 CloudWatch las funciones.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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 Kotlin code example performs the following tasks:

 1. List available namespaces from Amazon CloudWatch. Select a namespace from the
 list.
 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.
 */

val DASHES: String? = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson> <dashboardAdd>
 <settings> <metricImage>

        Where:
            myDate - The start date to use to get metric statistics. (For example,
 2023-01-11T18:35:24.00Z.)
            costDateWeek - The start date to use to get AWS Billing and Cost
 Management statistics. (For example, 2023-01-11T18:35:24.00Z.)
            dashboardName - The name of the dashboard to create.
            dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)
            dashboardAdd - The location of a JSON file to use to update a dashboard.
 (See Readme file.)
            settings - The location of a JSON file from which various values are
 read. (See Readme file.)
            metricImage - The location of a BMP file that is used to create a
 graph.
    """

    if (args.size != 7) {
        println(usage)
        System.exit(1)
    }

    val myDate = args[0]
    val costDateWeek = args[1]
    val dashboardName = args[2]
    val dashboardJson = args[3]
    val dashboardAdd = args[4]
    val settings = args[5]
    var metricImage = args[6]
    val dataPoint = "10.0".toDouble()
    val inOb = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the Amazon CloudWatch example scenario.")
```

```
    println(DASHES)

    println(DASHES)
    println("1. List at least five available unique namespaces from Amazon
CloudWatch. Select a CloudWatch namespace from the list.")
    val list: ArrayList<String> = listNameSpaces()
    for (z in 0..4) {
        println("    ${z + 1}. ${list[z]}")
    }

    var selectedNamespace: String
    var selectedMetrics = ""
    var num = inOb.nextLine().toInt()
    println("You selected $num")

    if (1 <= num && num <= 5) {
        selectedNamespace = list[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $selectedNamespace")
    println(DASHES)

    println(DASHES)
    println("2. List available metrics within the selected namespace and select one
from the list.")
    val metList = listMets(selectedNamespace)
    for (z in 0..4) {
        println("    ${ z + 1}. ${metList?.get(z)}")
    }
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        selectedMetrics = metList![num - 1]
    } else {
        println("You did not select a valid option.")
        System.exit(1)
    }
    println("You selected $selectedMetrics")
    val myDimension = getSpecificMet(selectedNamespace)
    if (myDimension == null) {
        println("Error - Dimension is null")
        exitProcess(1)
    }
```

```
    println(DASHES)

    println(DASHES)
    println("3. Get statistics for the selected metric over the last day.")
    val metricOption: String
    val statTypes = ArrayList<String>()
    statTypes.add("SampleCount")
    statTypes.add("Average")
    statTypes.add("Sum")
    statTypes.add("Minimum")
    statTypes.add("Maximum")

    for (t in 0..4) {
        println("    ${t + 1}. ${statTypes[t]}")
    }
    println("Select a metric statistic by entering a number from the preceding
list:")
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        metricOption = statTypes[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $metricOption")
    getAndDisplayMetricStatistics(selectedNamespace, selectedMetrics, metricOption,
myDate, myDimension)
    println(DASHES)

    println(DASHES)
    println("4. Get CloudWatch estimated billing for the last week.")
    getMetricStatistics(costDateWeek)
    println(DASHES)

    println(DASHES)
    println("5. Create a new CloudWatch dashboard with metrics.")
    createDashboardWithMetrics(dashboardName, dashboardJson)
    println(DASHES)

    println(DASHES)
    println("6. List dashboards using a paginator.")
    listDashboards()
    println(DASHES)
```

```
    println(DASHES)
    println("7. Create a new custom metric by adding data to it.")
    createNewCustomMetric(dataPoint)
    println(DASHES)

    println(DASHES)
    println("8. Add an additional metric to the dashboard.")
    addMetricToDashboard(dashboardAdd, dashboardName)
    println(DASHES)

    println(DASHES)
    println("9. Create an alarm for the custom metric.")
    val alarmName: String = createAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("10. Describe 10 current alarms.")
    describeAlarms()
    println(DASHES)

    println(DASHES)
    println("11. Get current data for the new custom metric.")
    getCustomMetricData(settings)
    println(DASHES)

    println(DASHES)
    println("12. Push data into the custom metric to trigger the alarm.")
    addMetricDataForAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("13. Check the alarm state using the action DescribeAlarmsForMetric.")
    checkForMetricAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("14. Get alarm history for the new alarm.")
    getAlarmHistory(settings, myDate)
    println(DASHES)

    println(DASHES)
    println("15. Add an anomaly detector for the custom metric.")
    addAnomalyDetector(settings)
    println(DASHES)
```

```
    println(DASHES)
    println("16. Describe current anomaly detectors.")
    describeAnomalyDetectors(settings)
    println(DASHES)

    println(DASHES)
    println("17. Get a metric image for the custom metric.")
    getAndOpenMetricImage(metricImage)
    println(DASHES)

    println(DASHES)
    println("18. Clean up the Amazon CloudWatch resources.")
    deleteDashboard(dashboardName)
    deleteAlarm(alarmName)
    deleteAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("The Amazon CloudWatch example scenario is complete.")
    println(DASHES)
}

suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
```

```
        println("Successfully deleted the Anomaly Detector.")
    }
}

suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}

suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}

suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
```

```
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}

suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}

suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
```

```
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}

suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }
```

```
    CloudWatchClient {
        credentialsProvider = EnvironmentCredentialsProvider()
        region = "us-east-1"
    }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}

suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
```

```
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}

suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request =
        PutMetricDataRequest {
```

```
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}

suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
```

```
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}

suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

```
suspend fun createAlarm(fileName: String): String {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode: JsonNode = ObjectMapper().readTree(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val emailTopic = rootNode.findValue("emailTopic").asText()
    val accountId = rootNode.findValue("accountId").asText()
    val region2 = rootNode.findValue("region").asText()

    // Create a List for alarm actions.
    val alarmActionObs: MutableList<String> = ArrayList()
    alarmActionObs.add("arn:aws:sns:$region2:$accountId:$emailTopic")
    val alarmRequest =
        PutMetricAlarmRequest {
            alarmActions = alarmActionObs
            alarmDescription = "Example metric alarm"
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanOrEqualToThreshold
            threshold = 100.00
            metricName = customMetricName
            namespace = customMetricNamespace
            evaluationPeriods = 1
            period = 10
            statistic = Statistic.Maximum
            datapointsToAlarm = 1
            treatMissingData = "ignore"
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(alarmRequest)
        println("$alarmNameVal was successfully created!")
        return alarmNameVal
    }
}

suspend fun addMetricToDashboard(
    fileNameVal: String,
    dashboardNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
```

```
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully updated.")
    }
}

suspend fun createNewCustomMetric(dataPoint: Double) {
    val dimension =
        Dimension {
            name = "UNIQUE_PAGES"
            value = "URLS"
        }

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = "PAGES_VISITED"
            unit = StandardUnit.None
            value = dataPoint
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
            dimensions = listOf(dimension)
        }

    val request =
        PutMetricDataRequest {
            namespace = "SITE/TRAFFIC"
            metricData = listOf(datum)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric PAGES_VISITED")
    }
}
```

```
suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}

suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}

fun readFileAsString(file: String): String =
 String(Files.readAllBytes(Paths.get(file)))

suspend fun getMetricStatistics(costDateWeek: String?) {
    val start = Instant.parse(costDateWeek)
    val endDate = Instant.now()
```

```
    val dimension =
        Dimension {
            name = "Currency"
            value = "USD"
        }

    val dimensionList: MutableList<Dimension> = ArrayList()
    dimensionList.add(dimension)

    val statisticsRequest =
        GetMetricStatisticsRequest {
            metricName = "EstimatedCharges"
            namespace = "AWS/Billing"
            dimensions = dimensionList
            statistics = listOf(Statistic.Maximum)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            period = 86400
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data: List<Datapoint>? = response.datapoints
        if (data != null) {
            if (!data.isEmpty()) {
                for (datapoint in data) {
                    println("Timestamp:  ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
```

```
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
```

```
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}

suspend fun getSpecificMet(namespaceVal: String?): Dimension? {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(request)
        val myList = response.metrics
        if (myList != null) {
            return myList[0].dimensions?.get(0)
        }
    }
    return null
}

suspend fun listNameSpaces(): ArrayList<String> {
    val nameSpaceList = ArrayList<String>()
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(ListMetricsRequest {})
        response.metrics?.forEach { metrics ->
            val data = metrics.namespace
            if (!nameSpaceList.contains(data)) {
                nameSpaceList.add(data!!)
            }
        }
    }
    return nameSpaceList
}
```

- Para API obtener más información, consulta los siguientes temas como APIreferencia sobre Kotlin.AWS SDK
  - DeleteAlarms

- DeleteAnomalyDetector
- DeleteDashboards
- DescribeAlarmHistory
- DescribeAlarms
- DescribeAlarmsForMetric
- DescribeAnomalyDetectors
- GetMetricData
- GetMetricStatistics
- GetMetricWidgetImage
- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

Acciones

`DeleteAlarms`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteAlarms`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
```

```
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- Para API obtener más información, consulta DeleteAlarmsla AWS SDKAPIreferencia sobre Kotlin.

`DeleteAnomalyDetector`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteAnomalyDetector`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
```

```
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- Para API obtener más información, consulta DeleteAnomalyDetectorla AWS SDKAPIreferencia sobre Kotlin.

`DeleteDashboards`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteDashboards`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- Para API obtener más información, consulta DeleteDashboardsla AWS SDKAPIreferencia sobre Kotlin.

`DescribeAlarmHistory`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeAlarmHistory`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient {
        credentialsProvider = EnvironmentCredentialsProvider()
        region = "us-east-1"
    }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
```

```
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}
```

- Para API obtener más información, consulta DescribeAlarmHistorylaAWS SDKAPIreferencia sobre Kotlin.

`DescribeAlarms`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeAlarms`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

- Para API obtener más información, consulta DescribeAlarmsla AWS SDKAPIreferencia sobre Kotlin.

`DescribeAlarmsForMetric`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeAlarmsForMetric`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
```

```
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}
```

- Para API obtener más información, consulta DescribeAlarmsForMetricla AWS SDKAPIreferencia sobre Kotlin.

`DescribeAnomalyDetectors`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeAnomalyDetectors`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
```

```
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}
```

- Para API obtener más información, consulta DescribeAnomalyDetectorsla AWS SDKAPIreferencia sobre Kotlin.

`DisableAlarmActions`

En el siguiente ejemplo de código, se muestra cómo utilizar `DisableAlarmActions`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun disableActions(alarmName: String) {
    val request =
        DisableAlarmActionsRequest {
            alarmNames = listOf(alarmName)
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- Para API obtener más información, consulta DisableAlarmActionsla AWS SDKAPIreferencia sobre Kotlin.

`EnableAlarmActions`

En el siguiente ejemplo de código, se muestra cómo utilizar `EnableAlarmActions`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun enableActions(alarm: String) {
    val request =
        EnableAlarmActionsRequest {
            alarmNames = listOf(alarm)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- Para API obtener más información, consulta EnableAlarmActionsla AWS SDKAPIreferencia sobre Kotlin.

`GetMetricData`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetMetricData`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
```

```
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}
```

- Para API obtener más información, consulta GetMetricDatala AWS SDKAPIreferencia sobre Kotlin.

`GetMetricStatistics`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetMetricStatistics`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
```

```
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

- Para API obtener más información, consulta GetMetricStatisticsla AWS SDKAPIreferencia sobre Kotlin.

`GetMetricWidgetImage`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetMetricWidgetImage`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}
```

- Para API obtener más información, consulta GetMetricWidgetImagela AWS SDKAPIreferencia sobre Kotlin.

`ListDashboards`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListDashboards`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

- Para API obtener más información, consulta ListDashboardsla AWS SDKAPIreferencia sobre Kotlin.

`ListMetrics`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListMetrics`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}
```

- Para API obtener más información, consulta ListMetricsla AWS SDKAPIreferencia sobre Kotlin.

`PutAnomalyDetector`

En el siguiente ejemplo de código, se muestra cómo utilizar `PutAnomalyDetector`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}
```

- Para API obtener más información, consulta PutAnomalyDetectorla AWS SDKAPIreferencia sobre Kotlin.

`PutDashboard`

En el siguiente ejemplo de código, se muestra cómo utilizar `PutDashboard`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}
```

- Para API obtener más información, consulta PutDashboardla AWS SDKAPIreferencia sobre Kotlin.

`PutMetricAlarm`

En el siguiente ejemplo de código, se muestra cómo utilizar `PutMetricAlarm`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun putMetricAlarm(
    alarmNameVal: String,
    instanceIdVal: String,
) {
    val dimensionOb =
        Dimension {
            name = "InstanceId"
            value = instanceIdVal
        }

    val request =
        PutMetricAlarmRequest {
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanThreshold
            evaluationPeriods = 1
            metricName = "CPUUtilization"
            namespace = "AWS/EC2"
            period = 60
            statistic = Statistic.fromValue("Average")
            threshold = 70.0
            actionsEnabled = false
            alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
 utilization exceeds 70%"
            unit = StandardUnit.fromValue("Seconds")
            dimensions = listOf(dimensionOb)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- Para API obtener más información, consulta PutMetricAlarmla AWS SDKAPIreferencia sobre Kotlin.

`PutMetricData`

En el siguiente ejemplo de código, se muestra cómo utilizar `PutMetricData`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)
```

```
    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- Para API obtener más información, consulta PutMetricDatala AWS SDKAPIreferencia sobre Kotlin.

CloudWatch Registra ejemplos que se utilizan SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin with CloudWatch Logs.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- Acciones

Acciones

`DeleteSubscriptionFilter`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteSubscriptionFilter`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- Para API obtener más información, consulta DeleteSubscriptionFilterla AWS SDKAPIreferencia sobre Kotlin.

`DescribeSubscriptionFilters`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeSubscriptionFilters`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient { region = "us-west-2" }.use { cwlClient ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- Para API obtener más información, consulta DescribeSubscriptionFilterslA AWS SDKAPIreferencia sobre Kotlin.

`StartLiveTail`

En el siguiente ejemplo de código, se muestra cómo utilizar `StartLiveTail`.

SDKpara Kotlin

Incluir los archivos requeridos.

```
import aws.sdk.kotlin.services.cloudwatchlogs.CloudWatchLogsClient
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailRequest
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailResponseStream
import kotlinx.coroutines.flow.takeWhile
```

Inicie la sesión de Live Tail.

```
    val client = CloudWatchLogsClient.fromEnvironment()

    val request = StartLiveTailRequest {
        logGroupIdentifiers = logGroupIdentifiersVal
        logStreamNames = logStreamNamesVal
```

```
        logEventFilterPattern = logEventFilterPatternVal
    }

    val startTime = System.currentTimeMillis()

    try {
        client.startLiveTail(request) { response ->
            val stream = response.responseStream
            if (stream != null) {
                /* Set a timeout to unsubcribe from the flow. This will:
                * 1). Close the stream
                * 2). Stop the Live Tail session
                */
                stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                    if (value is StartLiveTailResponseStream.SessionStart) {
                        println(value.asSessionStart())
                    } else if (value is StartLiveTailResponseStream.SessionUpdate) {
                        for (e in value.asSessionUpdate().sessionResults!!) {
                            println(e)
                        }
                    } else {
                        throw IllegalArgumentException("Unknown event type")
                    }
                }
            } else {
                throw IllegalArgumentException("No response stream")
            }
        }
    } catch (e: Exception) {
        println("Exception occurred during StartLiveTail: $e")
        System.exit(1)
    }
```

- Para API obtener más información, consulta StartLiveTaillla referencia AWS SDK sobre Kotlin API.

Ejemplos de proveedores de identidad de Amazon Cognito que se utilizan SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Acciones

`AdminGetUser`

En el siguiente ejemplo de código, se muestra cómo utilizar `AdminGetUser`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
```

```
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- Para API obtener más información, consulta AdminGetUserla AWS SDKAPIreferencia sobre Kotlin.

`AdminInitiateAuth`

En el siguiente ejemplo de código, se muestra cómo utilizar `AdminInitiateAuth`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
```

```
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- Para API obtener más información, consulta AdminInitiateAuthla AWS SDKAPIreferencia sobre Kotlin.

`AdminRespondToAuthChallenge`

En el siguiente ejemplo de código, se muestra cómo utilizar `AdminRespondToAuthChallenge`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
```

```
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- Para API obtener más información, consulta AdminRespondToAuthChallengela AWS SDKAPIreferencia sobre Kotlin.

`AssociateSoftwareToken`

En el siguiente ejemplo de código, se muestra cómo utilizar `AssociateSoftwareToken`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }
```

```
    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}
```

- Para API obtener más información, consulta AssociateSoftwareTokenla AWS SDKAPIreferencia sobre Kotlin.

`ConfirmSignUp`

En el siguiente ejemplo de código, se muestra cómo utilizar `ConfirmSignUp`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
```

```
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- Para API obtener más información, consulta ConfirmSignUpla AWS SDKAPIreferencia sobre Kotlin.

`ListUsers`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListUsers`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- Para API obtener más información, consulta ListUsersla AWS SDKAPIreferencia sobre Kotlin.

`ResendConfirmationCode`

En el siguiente ejemplo de código, se muestra cómo utilizar `ResendConfirmationCode`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- Para API obtener más información, consulta ResendConfirmationCodela AWS SDKAPIreferencia sobre Kotlin.

`SignUp`

En el siguiente ejemplo de código, se muestra cómo utilizar `SignUp`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- Para API obtener más información, consulta SignUpla AWS SDKAPIreferencia sobre Kotlin.

`VerifySoftwareToken`

En el siguiente ejemplo de código, se muestra cómo utilizar `VerifySoftwareToken`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- Para API obtener más información, consulta VerifySoftwareTokenla AWS SDKAPIreferencia sobre Kotlin.

Escenarios

Registra un usuario con 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 multifactorial asociando una MFA aplicación al usuario.

- Inicie sesión con una contraseña y un MFA código.

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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 initiateAuth 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.
 */

suspend fun main(args: Array<String>) {
    val 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.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a new
code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
    confirmSignUp(clientId, code, userName)
```

```
    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
```

```
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
```

```
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}

suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }
```

```
    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Ejemplos de Amazon Comprehend utilizando Kotlin SDK

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con Amazon Comprehend.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Escenarios

Crear una aplicación de mensajería

El siguiente ejemplo de código muestra cómo crear una aplicación de mensajería mediante AmazonSQS.

SDKpara Kotlin

> Muestra cómo usar Amazon SQS API para desarrollar un Spring REST API que envíe y recupere mensajes.
>
> Para obtener el código fuente completo e instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.
>
> Servicios utilizados en este ejemplo
>
> - Amazon Comprehend
> - Amazon SQS

Ejemplos de DynamoDB que utilizan para Kotlin SDK

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con DynamoDB.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Conceptos básicos

Conceptos básicos

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.
- Escriba los datos de la película en la tabla a partir de un archivo de ejemplo. JSON
- 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.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.

```
suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }
```

```
    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}
```

Cree una función auxiliar para descargar y extraer el JSON archivo de muestra.

```
// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
```

```
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}
```

Obtener un elemento de una tabla.

```
suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
```

```
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

Ejemplo completo.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
          <fileName>

        Where:
           fileName - The path to the moviedata.json you can download from the
 Amazon DynamoDB Developer Guide.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    // Get the moviedata.json from the Amazon DynamoDB Developer Guide.
    val tableName = "Movies"
    val fileName = args[0]
    val partitionAlias = "#a"

    println("Creating an Amazon DynamoDB table named Movies with a key named id and
 a sort key named title.")
    createScenarioTable(tableName, "year")
    loadData(tableName, fileName)
    getMovie(tableName, "year", "1933")
    scanMovies(tableName)
    val count = queryMovieTable(tableName, "year", partitionAlias)
    println("There are $count Movies released in 2013.")
    deletIssuesTable(tableName)
}

suspend fun createScenarioTable(
    tableNameVal: String,
```

```
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
```

```
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}

// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
```

```
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}

suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}

suspend fun deletIssuesTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }
```

```
    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun queryMovieTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = "year"

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.N("2013")

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}

suspend fun scanMovies(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
```

```
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Para API obtener más información, consulta los siguientes temas en AWS SDK la sección de referencia sobre Kotlin API.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Acciones

`CreateTable`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateTable`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createNewTable(
    tableNameVal: String,
    key: String,
```

```
): String? {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef)
            keySchema = listOf(keySchemaVal)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        var tableArn: String
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        tableArn = response.tableDescription!!.tableArn.toString()
        println("Table $tableArn is ready")
        return tableArn
    }
}
```

- Para API obtener más información, consulta CreateTablela AWS SDKAPIreferencia sobre Kotlin.

`DeleteItem`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteItem`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteDynamoDBItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        DeleteItemRequest {
            tableName = tableNameVal
            key = keyToGet
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- Para API obtener más información, consulta DeleteItemla AWS SDKAPIreferencia sobre Kotlin.

`DeleteTable`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteTable`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- Para API obtener más información, consulta DeleteTablela AWS SDKAPIreferencia sobre Kotlin.

`GetItem`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetItem`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getSpecificItem(
    tableNameVal: String,
    keyName: String,
```

```
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

- Para API obtener más información, consulta GetItemla AWS SDKAPIreferencia sobre Kotlin.

`ListTables`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListTables`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listAllTables() {
    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
            println("Table name is $tableName")
```

```
        }
    }
}
```

- Para API obtener más información, consulta ListTablesla AWS SDKAPIreferencia sobre Kotlin.

`PutItem`

En el siguiente ejemplo de código, se muestra cómo utilizar `PutItem`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun putItemInTable(
    tableNameVal: String,
    key: String,
    keyVal: String,
    albumTitle: String,
    albumTitleValue: String,
    awards: String,
    awardVal: String,
    songTitle: String,
    songTitleVal: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()

    // Add all content to the table.
    itemValues[key] = AttributeValue.S(keyVal)
    itemValues[songTitle] = AttributeValue.S(songTitleVal)
    itemValues[albumTitle] = AttributeValue.S(albumTitleValue)
    itemValues[awards] = AttributeValue.S(awardVal)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
```

```
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- Para API obtener más información, consulta PutItemla AWS SDKAPIreferencia sobre Kotlin.

`Query`

En el siguiente ejemplo de código, se muestra cómo utilizar `Query`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun queryDynTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionKeyVal: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = partitionKeyName

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.S(partitionKeyVal)

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
```

```
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

- Para API obtener más información, consulta la sección Consulta la AWS SDKAPIreferencia sobre Kotlin.

`Scan`

En el siguiente ejemplo de código, se muestra cómo utilizar `Scan`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun scanItems(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Para API obtener más información, consulta Escanear en AWS SDK busca de API referencia sobre Kotlin.

`UpdateItem`

En el siguiente ejemplo de código, se muestra cómo utilizar `UpdateItem`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun updateTableItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
    name: String,
    updateVal: String,
) {
    val itemKey = mutableMapOf<String, AttributeValue>()
    itemKey[keyName] = AttributeValue.S(keyVal)

    val updatedValues = mutableMapOf<String, AttributeValueUpdate>()
    updatedValues[name] =
        AttributeValueUpdate {
            value = AttributeValue.S(updateVal)
            action = AttributeAction.Put
        }

    val request =
        UpdateItemRequest {
            tableName = tableNameVal
            key = itemKey
            attributeUpdates = updatedValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
```

```
    }
}
```

- Para API obtener más información, consulta UpdateItemla AWS SDKAPIreferencia sobre Kotlin.

Escenarios

Creación de una aplicación para enviar datos a una tabla de DynamoDB

El siguiente ejemplo de código muestra cómo crear una aplicación que envíe datos a una tabla de Amazon DynamoDB y le notifique cuando un usuario actualice la tabla.

SDKpara Kotlin

Muestra cómo crear una aplicación nativa de Android que envíe datos mediante Amazon API DynamoDB Kotlin y envíe un mensaje de texto mediante Amazon Kotlin. SNS API

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 SNS

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDKpara Kotlin

Muestra cómo desarrollar una aplicación de administración 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

- Gateway de API
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Creación de una aplicación web para hacer un seguimiento de los datos de DynamoDB

El siguiente ejemplo de código muestra cómo crear una aplicación web que haga un seguimiento de los elementos de trabajo de una tabla de Amazon DynamoDB y utilice Amazon Simple Email Service (SESAmazon) para enviar informes.

SDKpara Kotlin

Muestra cómo utilizar Amazon DynamoDB para crear una aplicación web API dinámica que realice un seguimiento de los datos de trabajo de DynamoDB.

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 SES

Consultar una tabla mediante lotes de instrucciones PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtenga un lote de elementos mediante la ejecución de varias SELECT instrucciones.
- Agregue un lote de elementos mediante la ejecución de varias INSERT declaraciones.
- Actualice un lote de elementos mediante la ejecución de varias UPDATE declaraciones.
- Elimine un lote de elementos mediante la ejecución de varias DELETE declaraciones.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun main() {
    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQBatch"
    println("Creating an Amazon DynamoDB table named $tableName with a key named id
 and a sort key named title.")
    createTablePartiQLBatch(ddb, tableName, "year")
    putRecordBatch(ddb)
    updateTableItemBatchBatch(ddb)
    deleteItemsBatch(ddb)
    deleteTablePartiQLBatch(tableName)
}

suspend fun createTablePartiQLBatch(
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }
```

```
    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun putRecordBatch(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?, 'title' : ?,
 'info' : ?}"

    // Create three movies to add to the Amazon DynamoDB table.
    val parametersMovie1 = mutableListOf<AttributeValue>()
    parametersMovie1.add(AttributeValue.N("2022"))
    parametersMovie1.add(AttributeValue.S("My Movie 1"))
    parametersMovie1.add(AttributeValue.S("No Information"))

    val statementRequestMovie1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie1
```

```
        }

    // Set data for Movie 2.
    val parametersMovie2 = mutableListOf<AttributeValue>()
    parametersMovie2.add(AttributeValue.N("2022"))
    parametersMovie2.add(AttributeValue.S("My Movie 2"))
    parametersMovie2.add(AttributeValue.S("No Information"))

    val statementRequestMovie2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie2
        }

    // Set data for Movie 3.
    val parametersMovie3 = mutableListOf<AttributeValue>()
    parametersMovie3.add(AttributeValue.N("2022"))
    parametersMovie3.add(AttributeValue.S("My Movie 3"))
    parametersMovie3.add(AttributeValue.S("No Information"))

    val statementRequestMovie3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestMovie1)
    myBatchStatementList.add(statementRequestMovie2)
    myBatchStatementList.add(statementRequestMovie3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }
    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: " + response.toString())
    println("Added new movies using a batch command.")
}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
```

```
        "UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack' where year=? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Update record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Update record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: $response")
```

```
    println("Updated three movies using a batch command.")
    println("Items were updated!")
}

suspend fun deleteItemsBatch(ddb: DynamoDbClient) {
    // Specify three records to delete.
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))

    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Specify record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Specify record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
```

```
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    ddb.batchExecuteStatement(batchRequest)
    println("Deleted three movies using a batch command.")
}

suspend fun deleteTablePartiQLBatch(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- Para API obtener más información, consulta BatchExecuteStatementla AWS SDKAPIreferencia sobre Kotlin.

Consultar una tabla con PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtenga un elemento mediante la ejecución de una SELECT declaración.
- Agregue un elemento mediante la ejecución de una INSERT declaración.
- Actualice un elemento mediante la ejecución de una UPDATE declaración.
- Elimine un elemento mediante la ejecución de una DELETE declaración.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
          <fileName>

        Where:
           fileName - The path to the moviedata.json file You can download from the
 Amazon DynamoDB Developer Guide.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQ"

    // Get the moviedata.json from the Amazon DynamoDB Developer Guide.
    val fileName = args[0]
    println("Creating an Amazon DynamoDB table named MoviesPartiQ with a key named
 id and a sort key named title.")
    createTablePartiQL(ddb, tableName, "year")
    loadDataPartiQL(ddb, fileName)

    println("******* Getting data from the MoviesPartiQ table.")
    getMoviePartiQL(ddb)

    println("******* Putting a record into the MoviesPartiQ table.")
    putRecordPartiQL(ddb)

    println("******* Updating a record.")
    updateTableItemPartiQL(ddb)

    println("******* Querying the movies released in 2013.")
    queryTablePartiQL(ddb)

    println("******* Deleting the MoviesPartiQ table.")
    deleteTablePartiQL(tableName)
}

suspend fun createTablePartiQL(
    ddb: DynamoDbClient,
```

```
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
```

```
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun loadDataPartiQL(
    ddb: DynamoDbClient,
    fileName: String,
) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
    var t = 0

    while (iter.hasNext()) {
        if (t == 200) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
        parameters.add(AttributeValue.N(year.toString()))
        parameters.add(AttributeValue.S(title))
        parameters.add(AttributeValue.S(info))

        executeStatementPartiQL(ddb, sqlStatement, parameters)
        println("Added Movie $title")
        parameters.clear()
        t++
    }
}

suspend fun getMoviePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?"
    val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
    parameters.add(AttributeValue.N("2012"))
```

```
    parameters.add(AttributeValue.S("The Perks of Being a Wallflower"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun putRecordPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2020"))
    parameters.add(AttributeValue.S("My Movie"))
    parameters.add(AttributeValue.S("No Info"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Added new movie.")
}

suspend fun updateTableItemPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C.
 Cooper\",\"Ernest B. Schoedsack\' where year=? and title=?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    parameters.add(AttributeValue.S("The East"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Item was updated!")
}

// Query the table where the year is 2013.
suspend fun queryTablePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year = ?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun deleteTablePartiQL(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
```

```
    }
}

suspend fun executeStatementPartiQL(
    ddb: DynamoDbClient,
    statementVal: String,
    parametersVal: List<AttributeValue>,
): ExecuteStatementResponse {
    val request =
        ExecuteStatementRequest {
            statement = statementVal
            parameters = parametersVal
        }

    return ddb.executeStatement(request)
}
```

- Para API obtener más información, consulta ExecuteStatementla AWS SDKAPIreferencia sobre Kotlin.

EC2Ejemplos de Amazon que utilizan SDK Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con AmazonEC2.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Introducción

Hola Amazon EC2

Los siguientes ejemplos de código muestran cómo empezar a utilizar AmazonEC2.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->

        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- Para API obtener más información, consulta DescribeSecurityGroupsla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Conceptos básicos
- Acciones

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Cree un par de claves y un grupo de seguridad.

- Selecciona una Amazon Machine Image (AMI) y un tipo de instancia compatible y, a continuación, crea una instancia.
- Detenga y vuelva a iniciar la instancia.
- Asocie una dirección IP elástica a su instancia.
- Conéctate a tu instancia con los recursos SSH y, a continuación, límpialos.

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin 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.
 16. Deletes the security group.
 17. Deletes the key pair.
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <keyName> <fileName> <groupName> <groupDesc> <vpcId> <myIpAddress>

        Where:
            keyName - A key pair name (for example, TestKeyPair).
            fileName - A file name where the key information is written to.
            groupName - The name of the security group.
            groupDesc - The description of the security group.
            vpcId - A VPC ID. You can get this value from the AWS Management
 Console.
            myIpAddress - The IP address of your development machine.

"""

    if (args.size != 6) {
        println(usage)
        exitProcess(0)
    }

    val keyName = args[0]
    val fileName = args[1]
    val groupName = args[2]
    val groupDesc = args[3]
    val vpcId = args[4]
    val myIpAddress = args[5]
    var newInstanceId: String? = ""

    println(DASHES)
    println("Welcome to the Amazon EC2 example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an RSA key pair and save the private key material as a .pem
 file.")
    createKeyPairSc(keyName, fileName)
```

```
    println(DASHES)

    println(DASHES)
    println("2. List key pairs.")
    describeEC2KeysSc()
    println(DASHES)

    println(DASHES)
    println("3. Create a security group.")
    val groupId = createEC2SecurityGroupSc(groupName, groupDesc, vpcId, myIpAddress)
    println(DASHES)

    println(DASHES)
    println("4. Display security group info for the newly created security group.")
    describeSecurityGroupsSc(groupId.toString())
    println(DASHES)

    println(DASHES)
    println("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2 in the
name.")
    val instanceId = getParaValuesSc()
    if (instanceId == "") {
        println("The instance Id value isn't valid.")
        exitProcess(0)
    }
    println("The instance Id is $instanceId.")
    println(DASHES)

    println(DASHES)
    println("6. Get more information about an amzn2 image and return the AMI
value.")
    val amiValue = instanceId?.let { describeImageSc(it) }
    if (instanceId == "") {
        println("The instance Id value is invalid.")
        exitProcess(0)
    }
    println("The AMI value is $amiValue.")
    println(DASHES)

    println(DASHES)
    println("7. Get a list of instance types.")
    val instanceType = getInstanceTypesSc()
    println(DASHES)
```

```
    println(DASHES)
    println("8. Create an instance.")
    if (amiValue != null) {
        newInstanceId = runInstanceSc(instanceType, keyName, groupName, amiValue)
        println("The instance Id is $newInstanceId")
    }
    println(DASHES)

    println(DASHES)
    println("9. Display information about the running instance. ")
    var ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("10.  Stop the instance.")
    if (newInstanceId != null) {
        stopInstanceSc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("11.  Start the instance.")
    if (newInstanceId != null) {
        startInstanceSc(newInstanceId)
    }
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("12. Allocate an Elastic IP address and associate it with the
instance.")
    val allocationId = allocateAddressSc()
    println("The allocation Id value is $allocationId")
    val associationId = associateAddressSc(newInstanceId, allocationId)
    println("The associate Id value is $associationId")
    println(DASHES)

    println(DASHES)
    println("13. Describe the instance again.")
    ipAddress = describeEC2InstancesSc(newInstanceId)
```

```
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("14. Disassociate and release the Elastic IP address.")
    disassociateAddressSc(associationId)
    releaseEC2AddressSc(allocationId)
    println(DASHES)

    println(DASHES)
    println("15. Terminate the instance and use a waiter.")
    if (newInstanceId != null) {
        terminateEC2Sc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("16. Delete the security group.")
    if (groupId != null) {
        deleteEC2SecGroupSc(groupId)
    }
    println(DASHES)

    println(DASHES)
    println("17. Delete the key pair.")
    deleteKeysSc(keyName)
    println(DASHES)

    println(DASHES)
    println("You successfully completed the Amazon EC2 scenario.")
    println(DASHES)
}

suspend fun deleteKeysSc(keyPair: String) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}
```

```
suspend fun deleteEC2SecGroupSc(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted security group with Id $groupIdVal")
    }
}

suspend fun terminateEC2Sc(instanceIdVal: String) {
    val ti =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceIdVal)
        }
    println("Wait for the instance to terminate. This will take a few minutes.")
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitUntilInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}

suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}

suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
```

```
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}

suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}

suspend fun allocateAddressSc(): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        return allocateResponse.allocationId
    }
}

suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
```

```
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}

suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

suspend fun describeEC2InstancesSc(newInstanceId: String?): String {
    var pubAddress = ""
    var isRunning = false
    val request =
        DescribeInstancesRequest {
            instanceIds = listOf(newInstanceId.toString())
        }

    while (!isRunning) {
        Ec2Client { region = "us-west-2" }.use { ec2 ->
            val response = ec2.describeInstances(request)
            val state =
                response.reservations
                    ?.get(0)
                    ?.instances
                    ?.get(0)
                    ?.state
                    ?.name
```

```
                    ?. value
            if (state != null) {
                if (state.compareTo("running") == 0) {
                    println("Image id is
 ${response.reservations!!.get(0).instances?.get(0)?.imageId}")
                    println("Instance type is
 ${response.reservations!!.get(0).instances?.get(0)?.instanceType}")
                    println("Instance state is
 ${response.reservations!!.get(0).instances?.get(0)?.state}")
                    pubAddress =
                        response.reservations!!
                            .get(0)
                            .instances
                            ?.get(0)
                            ?.publicIpAddress
                            .toString()
                    println("Instance address is $pubAddress")
                    isRunning = true
                }
            }
        }
    }
    return pubAddress
}

suspend fun runInstanceSc(
    instanceTypeVal: String,
    keyNameVal: String,
    groupNameVal: String,
    amiIdVal: String,
): String {
    val runRequest =
        RunInstancesRequest {
            instanceType = InstanceType.fromValue(instanceTypeVal)
            keyName = keyNameVal
            securityGroups = listOf(groupNameVal)
            maxCount = 1
            minCount = 1
            imageId = amiIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(runRequest)
        val instanceId = response.instances?.get(0)?.instanceId
```

```
        println("Successfully started EC2 Instance $instanceId based on AMI
 $amiIdVal")
        return instanceId.toString()
    }
}

// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}

// Display the Description field that corresponds to the instance Id value.
suspend fun describeImageSc(instanceId: String): String? {
    val imagesRequest =
        DescribeImagesRequest {
            imageIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeImages(imagesRequest)
```

```
        println("The description of the first image is
 ${response.images?.get(0)?.description}")
        println("The name of the first image is  ${response.images?.get(0)?.name}")

        // Return the image Id value.
        return response.images?.get(0)?.imageId
    }
}

// Get the Id value of an instance with amzn2 in the name.
suspend fun getParaValuesSc(): String? {
    val parameterRequest =
        GetParametersByPathRequest {
            path = "/aws/service/ami-amazon-linux-latest"
        }

    SsmClient { region = "us-west-2" }.use { ssmClient ->
        val response = ssmClient.getParametersByPath(parameterRequest)
        response.parameters?.forEach { para ->
            println("The name of the para is: ${para.name}")
            println("The type of the para is: ${para.type}")
            println("")
            if (para.name?.let { filterName(it) } == true) {
                return para.value
            }
        }
    }
    return ""
}

fun filterName(name: String): Boolean {
    val parts = name.split("/").toTypedArray()
    val myValue = parts[4]
    return myValue.contains("amzn2")
}

suspend fun describeSecurityGroupsSc(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
```

```
        for (group in response.securityGroups!!) {
            println("Found Security Group with id " + group.groupId.toString() + "
 and group VPC " + group.vpcId)
        }
    }
}

suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
```

```
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}

suspend fun describeEC2KeysSc() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}

suspend fun createKeyPairSc(
    keyNameVal: String,
    fileNameVal: String,
) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        val content = response.keyMaterial
        if (content != null) {
            File(fileNameVal).writeText(content)
        }
        println("Successfully created key pair named $keyNameVal")
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - AllocateAddress

- AssociateAddress
- AuthorizeSecurityGroupIngress
- CreateKeyPair
- CreateSecurityGroup
- DeleteKeyPair
- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Acciones

`AllocateAddress`

En el siguiente ejemplo de código, se muestra cómo utilizar `AllocateAddress`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
```

```
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        val allocationIdVal = allocateResponse.allocationId

        val request =
            AssociateAddressRequest {
                instanceId = instanceIdVal
                allocationId = allocationIdVal
            }

        val associateResponse = ec2.associateAddress(request)
        return associateResponse.associationId
    }
}
```

- Para API obtener más información, consulta AllocateAddressla AWS SDKAPIreferencia sobre Kotlin.

`AssociateAddress`

En el siguiente ejemplo de código, se muestra cómo utilizar `AssociateAddress`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
```

```
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}
```

- Para API obtener más información, consulta AssociateAddressla AWS SDKAPIreferencia sobre Kotlin.

`AuthorizeSecurityGroupIngress`

En el siguiente ejemplo de código, se muestra cómo utilizar `AuthorizeSecurityGroupIngress`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
```

```
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}
```

- Para API obtener más información, consulta AuthorizeSecurityGroupIngressla AWS SDKAPIreferencia sobre Kotlin.

`CreateKeyPair`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateKeyPair`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- Para API obtener más información, consulta CreateKeyPairla AWS SDKAPIreferencia sobre Kotlin.

`CreateSecurityGroup`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateSecurityGroup`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
```

```
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "0.0.0.0/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}
```

- Para API obtener más información, consulta CreateSecurityGroupla AWS SDKAPIreferencia sobre Kotlin.

`DeleteKeyPair`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteKeyPair`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- Para API obtener más información, consulta DeleteKeyPairla AWS SDKAPIreferencia sobre Kotlin.

`DeleteSecurityGroup`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteSecurityGroup`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- Para API obtener más información, consulta DeleteSecurityGroupla AWS SDKAPIreferencia sobre Kotlin.

`DescribeInstanceTypes`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeInstanceTypes`.

SDKpara Kotlin

> **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 instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
```

```
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}
```

- Para API obtener más información, consulta DescribeInstanceTypesla AWS SDKAPIreferencia sobre Kotlin.

`DescribeInstances`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeInstances`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstances(request)
        response.reservations?.forEach { reservation ->
            reservation.instances?.forEach { instance ->
                println("Instance Id is ${instance.instanceId}")
                println("Image id is ${instance.imageId}")
                println("Instance type is ${instance.instanceType}")
                println("Instance state name is ${instance.state?.name}")
                println("monitoring information is ${instance.monitoring?.state}")
            }
        }
    }
}
```

- Para API obtener más información, consulta DescribeInstancesla AWS SDKAPIreferencia sobre Kotlin.

`DescribeKeyPairs`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeKeyPairs`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeEC2Keys() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
```

```
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}
```

- Para API obtener más información, consulta DescribeKeyPairsla AWS SDKAPIreferencia sobre Kotlin.

`DescribeSecurityGroups`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeSecurityGroups`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->

        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- Para API obtener más información, consulta DescribeSecurityGroupsla AWS SDKAPIreferencia sobre Kotlin.

`DisassociateAddress`

En el siguiente ejemplo de código, se muestra cómo utilizar `DisassociateAddress`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- Para API obtener más información, consulta DisassociateAddressla AWS SDKAPIreferencia sobre Kotlin.

`ReleaseAddress`

En el siguiente ejemplo de código, se muestra cómo utilizar `ReleaseAddress`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun releaseEC2AddressSc(allocId: String?) {
```

```
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- Para API obtener más información, consulta ReleaseAddressla AWS SDKAPIreferencia sobre Kotlin.

`RunInstances`

En el siguiente ejemplo de código, se muestra cómo utilizar `RunInstances`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createEC2Instance(
    name: String,
    amiId: String,
): String? {
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(request)
```

```
        val instanceId = response.instances?.get(0)?.instanceId
        val tag =
            Tag {
                key = "Name"
                value = name
            }

        val requestTags =
            CreateTagsRequest {
                resources = listOf(instanceId.toString())
                tags = listOf(tag)
            }
        ec2.createTags(requestTags)
        println("Successfully started EC2 Instance $instanceId based on AMI $amiId")
        return instanceId
    }
}
```

- Para API obtener más información, consulta RunInstancesla AWS SDKAPIreferencia sobre Kotlin.

`StartInstances`

En el siguiente ejemplo de código, se muestra cómo utilizar `StartInstances`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
```

```
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

- Para API obtener más información, consulta StartInstancesla AWS SDKAPIreferencia sobre Kotlin.

`StopInstances`

En el siguiente ejemplo de código, se muestra cómo utilizar `StopInstances`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
```

```
        println("Successfully stopped instance $instanceId")
    }
}
```

- Para API obtener más información, consulta StopInstancesla AWS SDKAPIreferencia sobre Kotlin.

`TerminateInstances`

En el siguiente ejemplo de código, se muestra cómo utilizar `TerminateInstances`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.terminateInstances(request)
        response.terminatingInstances?.forEach { instance ->
            println("The ID of the terminated instance is ${instance.instanceId}")
        }
    }
}
```

- Para API obtener más información, consulta TerminateInstancesla AWS SDKAPIreferencia sobre Kotlin.

ECREjemplos de Amazon que utilizan SDK Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con AmazonECR.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Introducción

Hola Amazon ECR

Los siguientes ejemplos de código muestran cómo empezar a utilizar AmazonECR.

SDKpara Kotlin

> **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 aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()
```

```
    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
        }
    }
}
```

- Para API obtener más información, consulta listImagesla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Conceptos básicos
- Acciones

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Crea un ECR repositorio de Amazon.
- Establezca políticas de repositorios.
- Recupere el repositorioURIs.
- Obtén los tokens de ECR autorización de Amazon.

- Establece políticas de ciclo de vida para los ECR repositorios de Amazon.
- Envía una imagen de Docker a un ECR repositorio de Amazon.
- Comprueba la existencia de una imagen en un ECR repositorio de Amazon.
- Haz una lista de ECR los repositorios de Amazon para tu cuenta y obtén información sobre ellos.
- Elimina los ECR repositorios de Amazon.

SDKpara Kotlin

> **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 ECR las funciones de Amazon.

```
import java.util.Scanner

/**
 * Before running this Kotlin code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 *
 * This 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 code example requires a local docker image named echo-text. Without a local
 image,
 * this program will not successfully run. For more information including how to
 create the local
 * image, see:
 *
 * /getting_started_scenarios/ecr_scenario/README
```

```
 *
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val 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.

        """.trimIndent()

    // if (args.size != 2) {
    //     println(usage)
    //     return
    // }

    var iamRole = "arn:aws:iam::814548047983:role/Admin"
    var localImageName: String
    var accountId = "814548047983"
    val ecrActions = ECRActions()
    val scanner = Scanner(System.`in`)

    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.

        The `EcrClient` service client that is part of the AWS SDK for Kotlin
 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).

        """.trimIndent(),
    )

    while (true) {
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("1", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else if (input.trim { it <= ' ' }.equals("2", ignoreCase = true)) {
            val repoName = "echo-text"
            ecrActions.deleteECRRepository(repoName)
            return
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }

    waitForInputToContinue(scanner)
    println(DASHES)
    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.

        """.trimIndent(),
    )

    // Ensure that a local docker image named echo-text exists.
    val doesExist = ecrActions.listLocalImages()
    val repoName: String
    if (!doesExist) {
        println("The local image named echo-text does not exist")
        return
    } else {
        localImageName = "echo-text"
        repoName = "echo-text"
    }

    val repoArn = ecrActions.createECRRepository(repoName).toString()
    println("The ARN of the ECR repository is $repoArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    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.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.setRepoPolicy(repoName, iamRole)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
```

```
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val policyText = ecrActions.getRepoPolicy(repoName)
    println("Policy Text:")
    println(policyText)
    waitForInputToContinue(scanner)

    println(DASHES)
    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.


        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.getAuthToken()
    waitForInputToContinue(scanner)

    println(DASHES)
    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.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val repositoryURI: String? = ecrActions.getRepositoryURI(repoName)
    println("The repository URI is $repositoryURI")
    waitForInputToContinue(scanner)

    println(DASHES)
    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.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val pol = ecrActions.setLifeCyclePolicy(repoName)
    println(pol)
    waitForInputToContinue(scanner)

    println(DASHES)
    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.

       """.trimIndent(),
   )

   waitForInputToContinue(scanner)
   ecrActions.pushDockerImage(repoName, localImageName)
   waitForInputToContinue(scanner)

   println(DASHES)
   println("8. Verify if the image is in the ECR Repository.")
   waitForInputToContinue(scanner)
   ecrActions.verifyImage(repoName, localImageName)
   waitForInputToContinue(scanner)

   println(DASHES)
   println("9. As an optional step, you can interact with the image in Amazon ECR
 by using the CLI.")
   println("Would you like to view instructions on how to use the CLI to run the
 image? (y/n)")
   val ans = scanner.nextLine().trim()
   if (ans.equals("y", true)) {
       val 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 $accountId.dkr.ecr.us-east-1.amazonaws.com

       2. Describe the image using this command:

          aws ecr describe-images --repository-name $repoName --image-ids imageTag=
$localImageName

       3. Run the Docker container and view the output using this command:
```

```
            docker run --rm $accountId.dkr.ecr.us-east-1.amazonaws.com/$repoName:
$localImageName
        """
        println(instructions)
    }
    waitForInputToContinue(scanner)

    println(DASHES)
    println("10. Delete the ECR Repository.")
    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.

        """.trimIndent(),
    )
    println("Would you like to delete the Amazon ECR Repository? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        println("You selected to delete the AWS ECR resources.")
        waitForInputToContinue(scanner)
        ecrActions.deleteECRRepository(repoName)
    }

    println(DASHES)
    println("This concludes the Amazon ECR SDK scenario")
    println(DASHES)
}

private fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
```

```
        }
    }
}
```

Una clase contenedora para los ECR SDK métodos de Amazon.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
import java.io.IOException
import java.util.Base64

class ECRActions {
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
 default port.
            val dockerCmdExecFactory: DockerCmdExecFactory =

 NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
            dockerClient =
 DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory).
        } else {
            dockerClient = DockerClientBuilder.getInstance().build()
        }
```

```
        return dockerClient
    }


    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
policy.
     */
    suspend fun setLifeCyclePolicy(repoName: String): String? {
        val polText =
            """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
            ]
            }

            """.trimIndent()
        val lifecyclePolicyPreviewRequest =
            StartLifecyclePolicyPreviewRequest {
                lifecyclePolicyText = polText
                repositoryName = repoName
            }

        // Execute the request asynchronously.
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
            return response.lifecyclePolicyText
        }
```

```
    }


    /**
     * 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.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }


    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
```

```
            }
        }
    }


    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }


    /**
     * 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.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version" : "2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
```

```
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }


    /**
     * 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 RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
```

```
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }

    suspend fun getRepoARN(repoName: String): String? {
        // Fetch the existing repository's ARN.
        val describeRequest =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeResponse = ecrClient.describeRepositories(describeRequest)
            return describeResponse.repositories?.get(0)?.repositoryArn
        }
    }

    fun listLocalImages(): Boolean = try {
        val images = getDockerClient()?.listImagesCmd()?.exec()
        images?.any { image ->
            image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
        } ?: false
    } catch (ex: Exception) {
        println("ERROR: ${ex.message}")
        false
    }


    /**
     * 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.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
```

```
        val authConfig = getAuthConfig(repoName)

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }


    /**
     * 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.
```

```
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }


    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
 empty")
        }

        val repositoryRequest =
```

```
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }

    // Return an AuthConfig.
    private suspend fun getAuthConfig(repoName: String): AuthConfig {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            val decodedToken = String(Base64.getDecoder().decode(token))
            val password = decodedToken.substring(4)

            val request =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val descrRepoResponse = ecrClient.describeRepositories(request)
            val repoData = descrRepoResponse.repositories?.firstOrNull
 { it.repositoryName == repoName }
            val registryURL: String = repoData?.repositoryUri?.split("/")?.get(0) ?:
 ""

            return AuthConfig()
                .withUsername("AWS")
                .withPassword(password)
                .withRegistryAddress(registryURL)
        }
    }
}
```

- Para API obtener más información, consulta los siguientes temas como referencia AWS SDK sobre Kotlin API.

- CreateRepository
- DeleteRepository
- DescribeImages
- DescribeRepositories
- GetAuthorizationToken
- GetRepositoryPolicy
- SetRepositoryPolicy
- StartLifecyclePolicyPreview

Acciones

`CreateRepository`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateRepository`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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 RepositoryAlreadyExistsException if the repository exists.
    * @throws EcrException         if an error occurs while creating the
repository.
    */
   suspend fun createECRRepository(repoName: String?): String? {
       val request =
           CreateRepositoryRequest {
               repositoryName = repoName
```

```
            }

        return try {
            EcrClient { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }
```

- Para API obtener más información, consulta CreateRepositorylа AWS SDKAPIreferencia sobre Kotlin.

`DeleteRepository`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteRepository`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.
 */
suspend fun deleteECRRepository(repoName: String) {
    if (repoName.isNullOrEmpty()) {
```

```
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- Para API obtener más información, consulta DeleteRepositoryla AWS SDKAPIreferencia sobre Kotlin.

`DescribeImages`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeImages`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.
     */
    suspend fun verifyImage(
```

```
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Para API obtener más información, consulta DescribeImagesla AWS SDKAPIreferencia sobre Kotlin.

`DescribeRepositories`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeRepositories`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }
```

- Para API obtener más información, consulta DescribeRepositoriesla AWS SDKAPIreferencia sobre Kotlin.

`GetAuthorizationToken`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetAuthorizationToken`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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).
     *
     */
    suspend fun getAuthToken() {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }
```

- Para API obtener más información, consulta GetAuthorizationTokenla AWS SDKAPIreferencia sobre Kotlin.

`GetRepositoryPolicy`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetRepositoryPolicy`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- Para API obtener más información, consulta GetRepositoryPolicyla AWS SDKAPIreferencia sobre Kotlin.

`PushImageCmd`

En el siguiente ejemplo de código, se muestra cómo utilizar `PushImageCmd`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.
    */
   suspend fun pushDockerImage(
       repoName: String,
       imageName: String,
   ) {
       println("Pushing $imageName to $repoName will take a few seconds")
       val authConfig = getAuthConfig(repoName)

       EcrClient { region = "us-east-1" }.use { ecrClient ->
           val desRequest =
               DescribeRepositoriesRequest {
                   repositoryNames = listOf(repoName)
               }

           val describeRepoResponse = ecrClient.describeRepositories(desRequest)
           val repoData =
               describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
                   ?: throw RuntimeException("Repository not found: $repoName")

           val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
           if (tagImageCmd != null) {
               tagImageCmd.exec()
           }
           val pushImageCmd =
               repoData.repositoryUri?.let {
```

```
                dockerClient?.pushImageCmd(it)
                    // ?.withTag("latest")
                    ?.withAuthConfig(authConfig)
            }

        try {
            if (pushImageCmd != null) {
                pushImageCmd.start().awaitCompletion()
            }
            println("The $imageName was pushed to Amazon ECR")
        } catch (e: IOException) {
            throw RuntimeException(e)
        }
    }
}
```

- Para API obtener más información, consulta PushImageCmdla AWS SDKAPIreferencia sobre Kotlin.

`SetRepositoryPolicy`

En el siguiente ejemplo de código, se muestra cómo utilizar `SetRepositoryPolicy`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.
 */
suspend fun setRepoPolicy(
    repoName: String?,
```

```
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version" : "2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- Para API obtener más información, consulta SetRepositoryPolicyla AWS SDKAPIreferencia sobre Kotlin.

`StartLifecyclePolicyPreview`

En el siguiente ejemplo de código, se muestra cómo utilizar `StartLifecyclePolicyPreview`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
```

```
            }
        }
    }
```

- Para API obtener más información, consulta StartLifecyclePolicyPreviewla AWS SDKAPIreferencia sobre Kotlin.

OpenSearch Ejemplos de servicios que se utilizan SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del servicio AWS SDK for Kotlin with OpenSearch .

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- Acciones

Acciones

`CreateDomain`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateDomain`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createNewDomain(domainNameVal: String?) {
```

```
    val clusterConfigOb =
        ClusterConfig {
            dedicatedMasterEnabled = true
            dedicatedMasterCount = 3
            dedicatedMasterType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceCount = 5
        }

    val ebsOptionsOb =
        EbsOptions {
            ebsEnabled = true
            volumeSize = 10
            volumeType = VolumeType.Gp2
        }

    val encryptionOptionsOb =
        NodeToNodeEncryptionOptions {
            enabled = true
        }

    val request =
        CreateDomainRequest {
            domainName = domainNameVal
            engineVersion = "OpenSearch_1.0"
            clusterConfig = clusterConfigOb
            ebsOptions = ebsOptionsOb
            nodeToNodeEncryptionOptions = encryptionOptionsOb
        }

    println("Sending domain creation request...")
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val createResponse = searchClient.createDomain(request)
        println("Domain status is ${createResponse.domainStatus}")
        println("Domain Id is ${createResponse.domainStatus?.domainId}")
    }
}
```

- Para API obtener más información, consulta CreateDomainla AWS SDKAPIreferencia sobre Kotlin.

`DeleteDomain`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteDomain`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteSpecificDomain(domainNameVal: String) {
    val request =
        DeleteDomainRequest {
            domainName = domainNameVal
        }
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        searchClient.deleteDomain(request)
        println("$domainNameVal was successfully deleted.")
    }
}
```

- Para API obtener más información, consulta DeleteDomainla AWS SDKAPIreferencia sobre Kotlin.

`ListDomainNames`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListDomainNames`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listAllDomains() {
```

```
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val response: ListDomainNamesResponse =
 searchClient.listDomainNames(ListDomainNamesRequest {})
        response.domainNames?.forEach { domain ->
            println("Domain name is " + domain.domainName)
        }
    }
}
```

- Para API obtener más información, consulta ListDomainNamesla AWS SDKAPIreferencia sobre Kotlin.

`UpdateDomainConfig`

En el siguiente ejemplo de código, se muestra cómo utilizar `UpdateDomainConfig`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            instanceCount = 3
        }

    val request =
        UpdateDomainConfigRequest {
            domainName = domainNameVal
            clusterConfig = clusterConfigOb
        }

    println("Sending domain update request...")
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val updateResponse = searchClient.updateDomainConfig(request)
        println("Domain update response from Amazon OpenSearch Service:")
```

```
        println(updateResponse.toString())
    }
}
```

- Para API obtener más información, consulta UpdateDomainConfigla AWS SDKAPIreferencia sobre Kotlin.

EventBridge ejemplos que utilizan SDK para Kotlin

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Kotlin with. EventBridge

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Introducción

Hola EventBridge

En los siguientes ejemplos de código se muestra cómo empezar a utilizar EventBridge.

SDKpara Kotlin

> **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 aws.sdk.kotlin.services.eventbridge.EventBridgeClient
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesRequest
```

```
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesResponse

suspend fun main() {
    listBusesHello()
}

suspend fun listBusesHello() {
    val request =
        ListEventBusesRequest {
            limit = 10
        }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val response: ListEventBusesResponse = eventBrClient.listEventBuses(request)
        response.eventBuses?.forEach { bus ->
            println("The name of the event bus is ${bus.name}")
            println("The ARN of the event bus is ${bus.arn}")
        }
    }
}
```

- Para API obtener más información, consulta ListEventBusesla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Conceptos básicos
- Acciones

Conceptos básicos

Conceptos básicos

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

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks with Amazon EventBridge:

 1. Creates an AWS Identity and Access Management (IAM) role to use with Amazon
 EventBridge.
 2. Creates an 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 S3 bucket.
 11. Disables a specific rule.
 12. Checks and prints 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 S3 bucket.
 16. Updates the rule to a custom rule pattern.
 17. Sends an event to trigger the rule.
 18. Cleans up resources.
*/
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
```

```
suspend fun main(args: Array<String>) {
    val 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.
    """
    val polJSON =
        "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
            "\"Action\": \"sts:AssumeRole\"" +
            "}]" +
            "}"

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val sc = Scanner(System.`in`)
    val roleName = args[0]
    val bucketName = args[1]
    val topicName = args[2]
    val eventRuleName = args[3]

    println(DASHES)
    println("Welcome to the Amazon EventBridge example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS Identity and Access Management (IAM) role to use with
 Amazon EventBridge.")
    val roleArn = createIAMRole(roleName, polJSON)
```

```
    println(DASHES)

    println(DASHES)
    println("2. Create an S3 bucket with EventBridge events enabled.")
    if (checkBucket(bucketName)) {
        println("$bucketName already exists. Ending this scenario.")
        exitProcess(1)
    }

    createBucket(bucketName)
    delay(3000)
    setBucketNotification(bucketName)
    println(DASHES)

    println(DASHES)
    println("3. Create a rule that triggers when an object is uploaded to Amazon
S3.")
    delay(10000)
    addEventRule(roleArn, bucketName, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("4. List rules on the event bus.")
    listRules()
    println(DASHES)

    println(DASHES)
    println("5. Create a new SNS topic for testing and let the user subscribe to the
topic.")
    val topicArn = createSnsTopic(topicName)
    println(DASHES)

    println(DASHES)
    println("6. Add a target to the rule that sends an email to the specified
topic.")
    println("Enter your email to subscribe to the Amazon SNS topic:")
    val email = sc.nextLine()
    subEmail(topicArn, email)
    println("Use the link in the email you received to confirm your subscription.
Then press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
```

```
    println("7. Create an EventBridge event that sends an email when an Amazon S3
object is created.")
    addSnsEventRule(eventRuleName, topicArn, topicName, eventRuleName, bucketName)
    println(DASHES)

    println(DASHES)
    println("8. List targets.")
    listTargets(eventRuleName)
    println(DASHES)

    println(DASHES)
    println(" 9. List the rules for the same target.")
    listTargetRules(topicArn)
    println(DASHES)

    println(DASHES)
    println("10. Trigger the rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("11. Disable a specific rule.")
    changeRuleState(eventRuleName, false)
    println(DASHES)

    println(DASHES)
    println("12. Check and print the state of the rule.")
    checkRule(eventRuleName)
    println(DASHES)

    println(DASHES)
    println("13. Add a transform to the rule to change the text of the email.")
    updateSnsEventRule(topicArn, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("14. Enable a specific rule.")
    changeRuleState(eventRuleName, true)
    println(DASHES)

    println(DASHES)
    println("15. Trigger the updated rule by uploading a file to the S3 bucket.")
```

```
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("16. Update the rule to a custom rule pattern.")
    updateToCustomRule(eventRuleName)
    println("Updated event rule $eventRuleName to use a custom pattern.")
    updateCustomRuleTargetWithTransform(topicArn, eventRuleName)
    println("Updated event target $topicArn.")
    println(DASHES)

    println(DASHES)
    println("17. Send an event to trigger the rule. This will trigger a subscription
 email.")
    triggerCustomRule(email)
    println("Events have been sent. Press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
    println("18. Clean up resources.")
    println("Do you want to clean up resources (y/n)")
    val ans = sc.nextLine()
    if (ans.compareTo("y") == 0) {
        cleanupResources(topicArn, eventRuleName, bucketName, roleName)
    } else {
        println("The resources will not be cleaned up. ")
    }
    println(DASHES)

    println(DASHES)
    println("The Amazon EventBridge example scenario has successfully completed.")
    println(DASHES)
}

suspend fun cleanupResources(
    topicArn: String?,
    eventRuleName: String?,
    bucketName: String?,
    roleName: String?,
) {
    println("Removing all targets from the event rule.")
```

```
    deleteTargetsFromRule(eventRuleName)
    deleteRuleByName(eventRuleName)
    deleteSNSTopic(topicArn)
    deleteS3Bucket(bucketName)
    deleteRole(roleName)
}

suspend fun deleteRole(roleNameVal: String?) {
    val policyArnVal = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    val policyRequest =
        DetachRolePolicyRequest {
            policyArn = policyArnVal
            roleName = roleNameVal
        }
    IamClient { region = "us-east-1" }.use { iam ->
        iam.detachRolePolicy(policyRequest)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")

        // Delete the role.
        val roleRequest =
            DeleteRoleRequest {
                roleName = roleNameVal
            }

        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteS3Bucket(bucketName: String?) {
    // Remove all the objects from the S3 bucket.
    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val myObjects = res.contents
        val toDelete = mutableListOf<ObjectIdentifier>()

        if (myObjects != null) {
            for (myValue in myObjects) {
                toDelete.add(
                    ObjectIdentifier {
```

```
                        key = myValue.key
                    },
                )
            }
        }

        val delOb =
            Delete {
                objects = toDelete
            }

        val dor =
            DeleteObjectsRequest {
                bucket = bucketName
                delete = delOb
            }
        s3Client.deleteObjects(dor)

        // Delete the S3 bucket.
        val deleteBucketRequest =
            DeleteBucketRequest {
                bucket = bucketName
            }
        s3Client.deleteBucket(deleteBucketRequest)
        println("You have deleted the bucket and the objects")
    }
}

// Delete the SNS topic.
suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
```

```
            name = ruleName
        }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}

suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}

suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
```

```
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}

suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}
```

```
suspend fun updateToCustomRule(ruleName: String?) {
    val customEventsPattern =
        "{" +
            "\"source\": [\"ExampleSource\"]," +
            "\"detail-type\": [\"ExampleType\"]" +
            "}"
    val request =
        PutRuleRequest {
            name = ruleName
            description = "Custom test rule"
            eventPattern = customEventsPattern
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putRule(request)
    }
}

// Update an Amazon S3 object created rule with a transform on the target.
suspend fun updateSnsEventRule(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()
    val myMap = mutableMapOf<String, String>()
    myMap["bucket"] = "$.detail.bucket.name"
    myMap["time"] = "$.time"

    val inputTransOb =
        InputTransformer {
            inputTemplate = "\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\""
            inputPathsMap = myMap
        }
    val targetOb =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransOb
        }

    val targetsRequest =
        PutTargetsRequest {
```

```
            rule = ruleName
            targets = listOf(targetOb)
            eventBusName = null
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}

suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
```

```
    }
}

// Create and upload a file to an S3 bucket to trigger an event.
@Throws(IOException::class)
suspend fun uploadTextFiletoS3(bucketName: String?) {
    val fileSuffix = SimpleDateFormat("yyyyMMddHHmmss").format(Date())
    val fileName = "TextFile$fileSuffix.txt"
    val myFile = File(fileName)
    val fw = FileWriter(myFile.absoluteFile)
    val bw = BufferedWriter(fw)
    bw.write("This is a sample file for testing uploads.")
    bw.close()

    val putOb =
        PutObjectRequest {
            bucket = bucketName
            key = fileName
            body = myFile.asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}

suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
```

```
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}

// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
}

suspend fun subEmail(
    topicArnVal: String?,
```

```
    email: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" Subscription ARN: ${result.subscriptionArn}")
    }
}

suspend fun createSnsTopic(topicName: String): String? {
    val topicPolicy =
        "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Sid\": \"EventBridgePublishTopic\"," +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
            "\"Resource\": \"*\"," +
            "\"Action\": \"sns:Publish\"" +
            "}]" +
            "}"

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest =
        CreateTopicRequest {
            name = topicName
            attributes = topicAttributes
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        println("Added topic $topicName for email subscriptions.")
        return response.topicArn
```

```
    }
}

suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}

// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """{
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
        "bucket": {
            "name": ["$bucketName"]
            }
        }
    }"""

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }
```

```
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}

// Set the Amazon S3 bucket notification configuration.
suspend fun setBucketNotification(bucketName: String) {
    val eventBridgeConfig =
        EventBridgeConfiguration {
        }

    val configuration =
        NotificationConfiguration {
            eventBridgeConfiguration = eventBridgeConfig
        }

    val configurationRequest =
        PutBucketNotificationConfigurationRequest {
            bucket = bucketName
            notificationConfiguration = configuration
            skipDestinationValidation = true
        }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putBucketNotificationConfiguration(configurationRequest)
        println("Added bucket $bucketName with EventBridge events enabled.")
    }
}

// Create an S3 bucket using a waiter.
suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        s3.waitUntilBucketExists {
            bucket = bucketName
        }
        println("$bucketName is ready")
    }
```

```
}

suspend fun checkBucket(bucketName: String?): Boolean {
    try {
        // Determine if the S3 bucket exists.
        val headBucketRequest =
            HeadBucketRequest {
                bucket = bucketName
            }

        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            return true
        }
    } catch (e: S3Exception) {
        System.err.println(e.message)
    }
    return false
}

suspend fun createIAMRole(
    rolenameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    val rolePolicyRequest =
        AttachRolePolicyRequest {
            roleName = rolenameVal
            policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
        }

    IamClient { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Acciones

`DeleteRule`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteRule`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
```

```
        println("Successfully deleted the rule")
    }
}
```

- Para API obtener más información, consulta DeleteRulela AWS SDKAPIreferencia sobre Kotlin.

`DescribeRule`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeRule`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

- Para API obtener más información, consulta DescribeRulela AWS SDKAPIreferencia sobre Kotlin.

`DisableRule`

En el siguiente ejemplo de código, se muestra cómo utilizar `DisableRule`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- Para API obtener más información, consulta DisableRulela AWS SDKAPIreferencia sobre Kotlin.

`EnableRule`

En el siguiente ejemplo de código, se muestra cómo utilizar `EnableRule`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- Para API obtener más información, consulta EnableRulela AWS SDKAPIreferencia sobre Kotlin.

`ListRuleNamesByTarget`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListRuleNamesByTarget`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}
```

- Para API obtener más información, consulta ListRuleNamesByTargetla AWS SDKAPIreferencia sobre Kotlin.

`ListRules`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListRules`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listRules() {
    val rulesRequest =
```

```
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}
```

- Para API obtener más información, consulta ListRuleslа AWS SDKAPIreferencia sobre Kotlin.

`ListTargetsByRule`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListTargetsByRule`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

- Para API obtener más información, consulta ListTargetsByRulela AWS SDKAPIreferencia sobre Kotlin.

`PutEvents`

En el siguiente ejemplo de código, se muestra cómo utilizar `PutEvents`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}
```

- Para API obtener más información, consulta PutEventsla AWS SDKAPIreferencia sobre Kotlin.

`PutRule`

En el siguiente ejemplo de código, se muestra cómo utilizar `PutRule`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Crear una regla programada.

```
suspend fun createScRule(
    ruleName: String?,
    cronExpression: String?,
) {
    val ruleRequest =
        PutRuleRequest {
            name = ruleName
            eventBusName = "default"
            scheduleExpression = cronExpression
            state = RuleState.Enabled
            description = "A test rule that runs on a schedule created by the Kotlin
 API"
        }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

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.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """{
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
        "bucket": {
            "name": ["$bucketName"]
            }
        }
    }"""

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- Para API obtener más información, consulta PutRulela AWS SDKAPIreferencia sobre Kotlin.

`PutTargets`

En el siguiente ejemplo de código, se muestra cómo utilizar `PutTargets`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
}
```

Agregue un transformador de entrada al destino de una regla.

```
suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- Para API obtener más información, consulta PutTargetsla AWS SDKAPIreferencia sobre Kotlin.

`RemoveTargets`

En el siguiente ejemplo de código, se muestra cómo utilizar `RemoveTargets`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}
```

- Para API obtener más información, consulta RemoveTargetsla AWS SDKAPIreferencia sobre Kotlin.

AWS Glue ejemplos que utilizan SDK para Kotlin

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Kotlin with. AWS Glue

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Cree un rastreador que rastree un bucket público de Amazon S3 y genere una base de datos de CSV metadatos con formato.
- Enumere la información sobre las bases de datos y las tablas en su. AWS Glue Data Catalog
- Cree una tarea para extraer CSV datos del depósito de S3, transformarlos y cargar la salida JSON con formato en otro depósito 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: Introducción a Studio. AWS Glue

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName> <scriptLocation>
 <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
 Management (IAM) role that has AWS Glue and Amazon Simple Storage Service (Amazon
 S3) permissions.
            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 (for example,
 cron(15 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a
 job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
    }

    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
    val jobName = args[5]
    val scriptLocation = args[6]
```

```
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(
    dbName: String?,
    locationUriVal: String?,
) {
    val input =
        DatabaseInput {
            description = "Built with the AWS SDK for Kotlin"
            name = dbName
            locationUri = locationUriVal
        }

    val request =
        CreateDatabaseRequest {
            databaseInput = input
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}

suspend fun createCrawler(
    iam: String?,
```

```
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val crawlerRequest =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
```

```
    }
}

suspend fun startCrawler(crawlerName: String) {
    val crawlerRequest =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {
    val tableRequest =
        GetTablesRequest {
            databaseName = dbName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}

suspend fun startJob(jobNameVal: String?) {
    val runRequest =
```

```
        StartJobRunRequest {
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(
    jobName: String,
    iam: String?,
    scriptLocationVal: String?,
) {
    val commandOb =
        JobCommand {
            pythonVersion = "3"
            name = "MyJob1"
            scriptLocation = scriptLocationVal
        }

    val jobRequest =
        CreateJobRequest {
            description = "A Job created by using the AWS SDK for Java V2"
            glueVersion = "2.0"
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            name = jobName
            role = iam
            command = commandOb
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
    val request =
        GetJobsRequest {
```

```
            maxResults = 10
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {
    val request =
        GetJobRunsRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {
    val jobRequest =
        DeleteJobRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {
    val request =
        DeleteDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
```

```
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
    val request =
        DeleteCrawlerRequest {
            name = crawlerName
        }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Acciones

`CreateCrawler`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateCrawler`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    // Add the S3Target to a list.
    val targetList = mutableListOf<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val request =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Kotlin API"
            targets = targetOb
            role = iam
```

```
            schedule = cron
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- Para API obtener más información, consulta CreateCrawlerla AWS SDKAPIreferencia sobre Kotlin.

`GetCrawler`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetCrawler`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getSpecificCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- Para API obtener más información, consulta GetCrawlerla AWS SDKAPIreferencia sobre Kotlin.

`GetDatabase`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetDatabase`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- Para API obtener más información, consulta GetDatabasela AWS SDKAPIreferencia sobre Kotlin.

`StartCrawler`

En el siguiente ejemplo de código, se muestra cómo utilizar `StartCrawler`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- Para API obtener más información, consulta StartCrawlerla AWS SDKAPIreferencia sobre Kotlin.

IAMejemplos que utilizan SDK para Kotlin

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Kotlin with. IAM

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- Conceptos básicos
- Acciones

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo crear un usuario y asumir un rol.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilices a IAM los usuarios para autenticarte cuando desarrolles software específico o trabajes 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.

SDKpara Kotlin

> **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 agrupen las acciones IAM del usuario.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
```

```
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }
```

```
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue: String =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
```

```
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
```

```
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient =
        StsClient {
            region = "us-east-1"
        }

    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials =
        StaticCredentialsProvider {
            accessKeyId = key
            secretAccessKey = secKey
            sessionToken = secToken
        }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 =
        S3Client {
```

```
            credentialsProvider = staticCredentials
            region = "us-east-1"
        }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
```

```
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- Para API obtener más información, consulta los siguientes temas como APIreferencia sobre Kotlin.AWS SDK
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser

- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Acciones

`AttachRolePolicy`

En el siguiente ejemplo de código, se muestra cómo utilizar `AttachRolePolicy`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }
```

```
        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

- Para API obtener más información, consulta AttachRolePolicyla AWS SDKAPIreferencia sobre Kotlin.

`CreateAccessKey`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateAccessKey`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- Para API obtener más información, consulta CreateAccessKeyla AWS SDKAPIreferencia sobre Kotlin.

`CreateAccountAlias`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateAccountAlias`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- Para API obtener más información, consulta CreateAccountAliasla AWS SDKAPIreferencia sobre Kotlin.

`CreatePolicy`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreatePolicy`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
```

```
        return response.policy?.arn.toString()
    }
}
```

- Para API obtener más información, consulta CreatePolicyla AWS SDKAPIreferencia sobre Kotlin.

`CreateUser`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateUser`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- Para API obtener más información, consulta CreateUserla AWS SDKAPIreferencia sobre Kotlin.

`DeleteAccessKey`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteAccessKey`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteKey(
    userNameVal: String,
    accessKey: String,
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- Para API obtener más información, consulta DeleteAccessKeyla AWS SDKAPIreferencia sobre Kotlin.

`DeleteAccountAlias`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteAccountAlias`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- Para API obtener más información, consulta DeleteAccountAliasla AWS SDKAPIreferencia sobre Kotlin.

`DeletePolicy`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeletePolicy`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- Para API obtener más información, consulta DeletePolicyla AWS SDKAPIreferencia sobre Kotlin.

`DeleteUser`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteUser`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteIAMUser(userNameVal: String) {
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- Para API obtener más información, consulta DeleteUserla AWS SDKAPIreferencia sobre Kotlin.

`DetachRolePolicy`

En el siguiente ejemplo de código, se muestra cómo utilizar `DetachRolePolicy`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
}
```

- Para API obtener más información, consulta DetachRolePolicyla AWS SDKAPIreferencia sobre Kotlin.

`GetPolicy`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetPolicy`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- Para API obtener más información, consulta GetPolicyla AWS SDKAPIreferencia sobre Kotlin.

`ListAccessKeys`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListAccessKeys`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- Para API obtener más información, consulta ListAccessKeysla AWS SDKAPIreferencia sobre Kotlin.

`ListAccountAliases`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListAccountAliases`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listAliases() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- Para API obtener más información, consulta ListAccountAliasesla AWS SDKAPIreferencia sobre Kotlin.

`ListUsers`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListUsers`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listAllUsers() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null) {
                println("Permissions boundary details
 ${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- Para API obtener más información, consulta ListUsersla AWS SDKAPIreferencia sobre Kotlin.

`UpdateUser`

En el siguiente ejemplo de código, se muestra cómo utilizar `UpdateUser`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
```

```
        println("Successfully updated user to $newName")
    }
}
```

- Para API obtener más información, consulta UpdateUserla AWS SDKAPIreferencia sobre Kotlin.

AWS IoT ejemplos que utilizan SDK para Kotlin

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Kotlin with. AWS IoT

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Introducción

Hola AWS IoT

En los siguientes ejemplos de código se muestra cómo empezar a utilizar AWS IoT.

SDKpara Kotlin

> **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 aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest
```

```
suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- Para API obtener más información, consulta listThingsla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Conceptos básicos
- Acciones

Conceptos básicos

Conceptos básicos

El siguiente ejemplo de código muestra cómo trabajar con la administración de AWS IoT dispositivos.

SDKpara Kotlin

> **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 aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
```

```
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-getting-
started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val 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.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    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 Kotlin 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 Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that can
be associated with a physical device.
        """.trimIndent(),
    )
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
    println("2. Generate a device certificate.")
    println(
        """
        A device certificate performs a role in securing the communication between
devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
    var certificateArn: String? = ""
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
```

```
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    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.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)

    println(DASHES)
    println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
    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.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    val endpointUrl = describeEndpoint()
    println(DASHES)

    println(DASHES)
    println("5. List your AWS IoT certificates")
    print("Press Enter to continue...")
    scanner.nextLine()
    if (certificateArn!!.isNotEmpty()) {
        listCertificates()
    } else {
        println("You did not create a certificates. Skipping this step.")
    }
    println(DASHES)
```

```
   println(DASHES)
   println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
   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.

       """.trimIndent(),
   )
   print("Press Enter to continue...")
   scanner.nextLine()
   updateShawdowThing(thingName)
   println(DASHES)

   println(DASHES)
   println("7. Write out the state information, in JSON format.")
   print("Press Enter to continue...")
   scanner.nextLine()
   getPayload(thingName)
   println(DASHES)

   println(DASHES)
   println("8. Creates a rule")
   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.
       """.trimIndent(),
   )
   print("Enter Rule name: ")
   val ruleName = scanner.nextLine()
   createIoTRule(roleARN, ruleName, snsAction)
   println(DASHES)

   println(DASHES)
   println("9. List your rules.")
   print("Press Enter to continue...")
```

```
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName? (y/
n)")
        val delAns = scanner.nextLine()
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
 true)) {
            println("11. You selected to detach amd delete the certificate.")
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
        println("11. You did not create a certificate so there is nothing to
 delete.")
    }
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase = true))
 {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)
```

```
    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
 { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
```

```
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient { region = "us-east-1" }.use { iotClient ->
        val listTopicRulesResponse = iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("--------------")
        }
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
```

```
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
```

```
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
}
```

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}
```

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}

suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
```

```
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

Acciones

`AttachThingPrincipal`

En el siguiente ejemplo de código, se muestra cómo utilizar `AttachThingPrincipal`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
```

```
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- Para API obtener más información, consulta AttachThingPrincipalla AWS SDKAPIreferencia sobre Kotlin.

`CreateKeysAndCertificate`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateKeysAndCertificate`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createCertificate(): String? {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}
```

- Para API obtener más información, consulta CreateKeysAndCertificatela AWS SDKAPIreferencia sobre Kotlin.

`CreateThing`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateThing`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Para API obtener más información, consulta CreateThingla AWS SDKAPIreferencia sobre Kotlin.

`CreateTopicRule`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateTopicRule`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- Para API obtener más información, consulta CreateTopicRulela AWS SDKAPIreferencia sobre Kotlin.

`DeleteCertificate`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteCertificate`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- Para API obtener más información, consulta DeleteCertificatela AWS SDKAPIreferencia sobre Kotlin.

`DeleteThing`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteThing`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
```

```
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- Para API obtener más información, consulta DeleteThingla AWS SDKAPIreferencia sobre Kotlin.

`DescribeEndpoint`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeEndpoint`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}
```

- Para API obtener más información, consulta DescribeEndpointla AWS SDKAPIreferencia sobre Kotlin.

`DescribeThing`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeThing`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- Para API obtener más información, consulta DescribeThingla AWS SDKAPIreferencia sobre Kotlin.

`DetachThingPrincipal`

En el siguiente ejemplo de código, se muestra cómo utilizar `DetachThingPrincipal`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- Para API obtener más información, consulta DetachThingPrincipalla AWS SDKAPIreferencia sobre Kotlin.

`ListCertificates`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListCertificates`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listCertificates() {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- Para API obtener más información, consulta ListCertificatesla AWS SDKAPIreferencia sobre Kotlin.

`SearchIndex`

En el siguiente ejemplo de código, se muestra cómo utilizar `SearchIndex`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
```

```
        }
    }
}
```

- Para API obtener más información, consulta SearchIndexla AWS SDKAPIreferencia sobre Kotlin.

`UpdateThing`

En el siguiente ejemplo de código, se muestra cómo utilizar `UpdateThing`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
```

```
            println("$thingNameVal attributes updated successfully.")
        }
    }
```

- Para API obtener más información, consulta UpdateThingla AWS SDKAPIreferencia sobre Kotlin.

AWS IoT data ejemplos que utilizan SDK para Kotlin

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Kotlin 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- Acciones

Acciones

`GetThingShadow`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetThingShadow`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
```

```
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}
```

- Para API obtener más información, consulta GetThingShadowla AWS SDKAPIreferencia sobre Kotlin.

`UpdateThingShadow`

En el siguiente ejemplo de código, se muestra cómo utilizar `UpdateThingShadow`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }
```

```
    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}
```

- Para API obtener más información, consulta UpdateThingShadowla AWS SDKAPIreferencia sobre Kotlin.

Ejemplos de Amazon Keyspaces que se utilizan SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con Amazon Keyspaces.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Introducción

Introducción a Amazon Keyspaces

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon Keyspaces.

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
*/

suspend fun main() {
    listKeyspaces()
}

suspend fun listKeyspaces() {
    val keyspacesRequest =
        ListKeyspacesRequest {
            maxResults = 10
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- Para API obtener más información, consulta ListKeyspacesla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Conceptos básicos
- Acciones

Conceptos básicos

Conceptos básicos

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.
- Conéctese al espacio de claves mediante una TLS conexión 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.

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This example uses a secure file format to hold certificate information for
 Kotlin 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 Kotlin 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.
 */

/*
   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.
 */
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main() {
    val fileName = "<Replace with the JSON file that contains movie data>"
    val keyspaceName = "<Replace with the name of the keyspace to create>"
    val titleUpdate = "The Family"
    val yearUpdate = 2013
    val tableName = "MovieKotlin"
    val tableNameRestore = "MovieRestore"

    val loader = DriverConfigLoader.fromClasspath("application.conf")
    val session =
        CqlSession
            .builder()
            .withConfigLoader(loader)
            .build()

    println(DASHES)
    println("Welcome to the Amazon Keyspaces example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create a keyspace.")
    createKeySpace(keyspaceName)
    println(DASHES)
```

```
    println(DASHES)
    delay(5000)
    println("2. Check for keyspace existence.")
    checkKeyspaceExistence(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("3. List keyspaces using a paginator.")
    listKeyspacesPaginator()
    println(DASHES)

    println(DASHES)
    println("4. Create a table with a simple movie data schema and enable point-in-
time recovery.")
    createTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("5. Check for the table to be in an Active state.")
    delay(6000)
    checkTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("6. List all tables in the keyspace.")
    listTables(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("7. Use a Cassandra driver to insert some records into the Movie
 table.")
    delay(6000)
    loadData(session, fileName, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("8. Get all records from the Movie table.")
    getMovieData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("9. Get a specific Movie.")
    getSpecificMovie(session, keyspaceName)
```

```
    println(DASHES)

    println(DASHES)
    println("10. Get a UTC timestamp for the current time.")
    val utc = ZonedDateTime.now(ZoneOffset.UTC)
    println("DATETIME = ${Date.from(utc.toInstant())}")
    println(DASHES)

    println(DASHES)
    println("11. Update the table schema to add a watched Boolean column.")
    updateTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("12. Update an item as watched.")
    delay(10000) // Wait 10 seconds for the update.
    updateRecord(session, keyspaceName, titleUpdate, yearUpdate)
    println(DASHES)

    println(DASHES)
    println("13. Query for items with watched = True.")
    getWatchedData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("14. Restore the table back to the previous state using the timestamp.")
    println("Note that the restore operation can take up to 20 minutes.")
    restoreTable(keyspaceName, utc)
    println(DASHES)

    println(DASHES)
    println("15. Check for completion of the restore action.")
    delay(5000)
    checkRestoredTable(keyspaceName, "MovieRestore")
    println(DASHES)

    println(DASHES)
    println("16. Delete both tables.")
    deleteTable(keyspaceName, tableName)
    deleteTable(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("17. Confirm that both tables are deleted.")
```

```
    checkTableDelete(keyspaceName, tableName)
    checkTableDelete(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("18. Delete the keyspace.")
    deleteKeyspace(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("The scenario has completed successfully.")
    println(DASHES)
}

suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}

suspend fun checkTableDelete(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var status: String
    var response: GetTableResponse
    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    try {
        KeyspacesClient { region = "us-east-1" }.use { keyClient ->
            // Keep looping until the table cannot be found and a
 ResourceNotFoundException is thrown.
            while (true) {
                response = keyClient.getTable(tableRequest)
                status = response.status.toString()
```

```
                println(". The table status is $status")
                delay(500)
            }
        }
    } catch (e: ResourceNotFoundException) {
        println(e.message)
    }
    println("The table is deleted")
}

suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}

suspend fun checkRestoredTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println("The table status is $status")
```

```
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }

        val cols = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}

fun getWatchedData(
    session: CqlSession,
    keyspaceName: String,
) {
```

```
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\"
 WHERE watched = true ALLOW FILTERING;")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

fun updateRecord(
    session: CqlSession,
    keySpace: String,
    titleUpdate: String?,
    yearUpdate: Int,
) {
    val sqlStatement =
        "UPDATE \"$keySpace\".\"MovieKotlin\" SET watched=true WHERE title = :k0 AND
 year = :k1;"
    val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    val preparedStatement = session.prepare(sqlStatement)
    builder.addStatement(
        preparedStatement
            .boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build(),
    )
    val batchStatement = builder.build()
    session.execute(batchStatement)
}

suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
```

```
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}

fun getSpecificMovie(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet =
        session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\" WHERE title
 = 'The Family' ALLOW FILTERING ;")

    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Get records from the Movie table.
fun getMovieData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin
\";")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Load data into the table.
fun loadData(
    session: CqlSession,
    fileName: String,
    keySpace: String,
```

```
) {
    val sqlStatement =
        "INSERT INTO \"$keySpace\".\"MovieKotlin\" (title, year, plot) values
 (:k0, :k1, :k2)"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        // Insert the data into the Amazon Keyspaces table.
        val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
        val preparedStatement: PreparedStatement = session.prepare(sqlStatement)
        builder.addStatement(
            preparedStatement
                .boundStatementBuilder()
                .setString("k0", title)
                .setInt("k1", year)
                .setString("k2", info)
                .build(),
        )

        val batchStatement = builder.build()
        session.execute(batchStatement)
        t++
    }
}

suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }
```

```
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}

suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}
```

```
suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
```

```
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}

suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
```

```
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}

suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Acciones

`CreateKeyspace`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateKeyspace`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Para API obtener más información, consulta CreateKeyspacela AWS SDKAPIreferencia sobre Kotlin.

`CreateTable`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateTable`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
```

```
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}
```

- Para API obtener más información, consulta CreateTablela AWS SDKAPIreferencia sobre Kotlin.

`DeleteKeyspace`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteKeyspace`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- Para API obtener más información, consulta DeleteKeyspacela AWS SDKAPIreferencia sobre Kotlin.

`DeleteTable`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteTable`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
```

```
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- Para API obtener más información, consulta DeleteTablela AWS SDKAPIreferencia sobre Kotlin.

`GetKeyspace`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetKeyspace`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}
```

- Para API obtener más información, consulta GetKeyspacela AWS SDKAPIreferencia sobre Kotlin.

`GetTable`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetTable`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
```

```
}
```

- Para API obtener más información, consulta GetTablela AWS SDKAPIreferencia sobre Kotlin.

`ListKeyspaces`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListKeyspaces`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}
```

- Para API obtener más información, consulta ListKeyspacesla AWS SDKAPIreferencia sobre Kotlin.

`ListTables`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListTables`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}
```

- Para API obtener más información, consulta ListTablesla AWS SDKAPIreferencia sobre Kotlin.

`RestoreTable`

En el siguiente ejemplo de código, se muestra cómo utilizar `RestoreTable`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun restoreTable(
```

```
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- Para API obtener más información, consulta RestoreTablela AWS SDKAPIreferencia sobre Kotlin.

`UpdateTable`

En el siguiente ejemplo de código, se muestra cómo utilizar `UpdateTable`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
```

```
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}
```

- Para API obtener más información, consulta UpdateTablela AWS SDKAPIreferencia sobre Kotlin.

AWS KMS ejemplos que utilizan SDK para Kotlin

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Kotlin 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- Acciones

Acciones

`CreateAlias`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateAlias`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- Para API obtener más información, consulta CreateAliasla AWS SDKAPIreferencia sobre Kotlin.

`CreateGrant`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateGrant`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createNewGrant(
    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationOb = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationOb)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
            operations = grantOperationList
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

- Para API obtener más información, consulta CreateGrantla AWS SDKAPIreferencia sobre Kotlin.

`CreateKey`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateKey`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- Para API obtener más información, consulta CreateKeyla AWS SDKAPIreferencia sobre Kotlin.

`Decrypt`

En el siguiente ejemplo de código, se muestra cómo utilizar `Decrypt`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
```

```
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
    path: String,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Write the decrypted data to a file.
        if (myVal != null) {
            File(path).writeBytes(myVal)
        }
    }
}
```

- Para API obtener más información, consulta Cómo descifrar AWS SDKpara obtener información sobre Kotlin API.

`DescribeKey`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeKey`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- Para API obtener más información, consulta DescribeKeyla AWS SDKAPIreferencia sobre Kotlin.

`DisableKey`

En el siguiente ejemplo de código, se muestra cómo utilizar `DisableKey`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- Para API obtener más información, consulta DisableKeyla AWS SDKAPIreferencia sobre Kotlin.

`EnableKey`

En el siguiente ejemplo de código, se muestra cómo utilizar `EnableKey`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- Para API obtener más información, consulta EnableKeyla AWS SDKAPIreferencia sobre Kotlin.

`Encrypt`

En el siguiente ejemplo de código, se muestra cómo utilizar `Encrypt`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
    path: String,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
```

```
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Write the decrypted data to a file.
        if (myVal != null) {
            File(path).writeBytes(myVal)
        }
    }
}
```

- Para API obtener más información, consulta Cómo cifrar como referencia AWS SDK sobre Kotlin API.

`ListAliases`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListAliases`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
            limit = 15
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- Para API obtener más información, consulta ListAliasesla AWS SDKAPIreferencia sobre Kotlin.

`ListGrants`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListGrants`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- Para API obtener más información, consulta ListGrantsla AWS SDKAPIreferencia sobre Kotlin.

`ListKeys`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListKeys`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listAllKeys() {
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- Para API obtener más información, consulta ListKeysla AWS SDKAPIreferencia sobre Kotlin.

Ejemplos de Lambda que se utilizan SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con Lambda.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- Conceptos básicos
- Acciones
- Escenarios

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Cree un IAM rol y una función Lambda y, a continuación, cargue el código del 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.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <functionName> <role> <handler> <bucketName> <updatedBucketName> <key>

        Where:
```

```
            functionName - The name of the AWS Lambda function.
            role - The AWS Identity and Access Management (IAM) service role that
has AWS Lambda permissions.
            handler - The fully qualified method name (for example,
example.Handler::handleRequest).
            bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
that contains the ZIP or JAR used for the Lambda function's code.
            updatedBucketName - The Amazon S3 bucket name that contains the .zip
or .jar used to update the Lambda function's code.
            key - The Amazon S3 key name that represents the .zip or .jar file (for
example, LambdaHello-1.0-SNAPSHOT.jar).
            """

   if (args.size != 6) {
       println(usage)
       exitProcess(1)
   }

   val functionName = args[0]
   val role = args[1]
   val handler = args[2]
   val bucketName = args[3]
   val updatedBucketName = args[4]
   val key = args[5]

   println("Creating a Lambda function named $functionName.")
   val funArn = createScFunction(functionName, bucketName, key, handler, role)
   println("The AWS Lambda ARN is $funArn")

   // Get a specific Lambda function.
   println("Getting the $functionName AWS Lambda function.")
   getFunction(functionName)

   // List the Lambda functions.
   println("Listing all AWS Lambda functions.")
   listFunctionsSc()

   // Invoke the Lambda function.
   println("*** Invoke the Lambda function.")
   invokeFunctionSc(functionName)

   // Update the AWS Lambda function code.
   println("*** Update the Lambda function code.")
   updateFunctionCode(functionName, updatedBucketName, key)
```

```
    // println("*** Invoke the function again after updating the code.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function configuration.
    println("Update the run time of the function.")
    updateFunctionConfiguration(functionName, handler)

    // Delete the AWS Lambda function.
    println("Delete the AWS Lambda function.")
    delFunction(functionName)
}

suspend fun createScFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java8
        }

    // Create a Lambda function using a waiter
    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn.toString()
    }
```

```
}

suspend fun getFunction(functionNameVal: String) {
    val functionRequest =
        GetFunctionRequest {
            functionName = functionNameVal
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val response = awsLambda.getFunction(functionRequest)
        println("The runtime of this Lambda function is
 ${response.configuration?.runtime}")
    }
}

suspend fun listFunctionsSc() {
    val request =
        ListFunctionsRequest {
            maxItems = 10
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val response = awsLambda.listFunctions(request)
        response.functions?.forEach { function ->
            println("The function name is ${function.functionName}")
        }
    }
}

suspend fun invokeFunctionSc(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            payload = byteArray
            logType = LogType.Tail
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("The function payload is ${res.payload?.toString(Charsets.UTF_8)}")
    }
}
```

```
suspend fun updateFunctionCode(
    functionNameVal: String?,
    bucketName: String?,
    key: String?,
) {
    val functionCodeRequest =
        UpdateFunctionCodeRequest {
            functionName = functionNameVal
            publish = true
            s3Bucket = bucketName
            s3Key = key
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val response = awsLambda.updateFunctionCode(functionCodeRequest)
        awsLambda.waitUntilFunctionUpdated {
            functionName = functionNameVal
        }
        println("The last modified value is " + response.lastModified)
    }
}

suspend fun updateFunctionConfiguration(
    functionNameVal: String?,
    handlerVal: String?,
) {
    val configurationRequest =
        UpdateFunctionConfigurationRequest {
            functionName = functionNameVal
            handler = handlerVal
            runtime = Runtime.Java11
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.updateFunctionConfiguration(configurationRequest)
    }
}

suspend fun delFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }
```

```
    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Acciones

`CreateFunction`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateFunction`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createNewFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
```

```
): String? {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java8
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn
    }
}
```

- Para API obtener más información, consulta CreateFunctionla AWS SDKAPIreferencia sobre Kotlin.

`DeleteFunction`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteFunction`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Para API obtener más información, consulta DeleteFunctionla AWS SDKAPIreferencia sobre Kotlin.

`Invoke`

En el siguiente ejemplo de código, se muestra cómo utilizar `Invoke`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun invokeFunction(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            logType = LogType.Tail
            payload = byteArray
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("${res.payload?.toString(Charsets.UTF_8)}")
```

```
        println("The log result is ${res.logResult}")
    }
}
```

- Para API obtener más información, consulta Invoke in como referencia AWS SDK sobre Kotlin API.

Escenarios

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDKpara Kotlin

Muestra cómo desarrollar una aplicación de administración 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

- Gateway de API
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

MediaConvert ejemplos que se utilizan SDK para Kotlin

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Kotlin 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Acciones

`CreateJob`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateJob`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createMediaJob(
    mcClient: MediaConvertClient,
    mcRoleARN: String,
    fileInputVal: String,
): String? {
    val s3path = fileInputVal.substring(0, fileInputVal.lastIndexOf('/') + 1) +
 "javasdk/out/"
    val fileOutput = s3path + "index"
    val thumbsOutput = s3path + "thumbs/"
    val mp4Output = s3path + "mp4/"

    try {
        val describeEndpoints =
            DescribeEndpointsRequest {
                maxResults = 20
            }
```

```
        val res = mcClient.describeEndpoints(describeEndpoints)
        if (res.endpoints?.size!! <= 0) {
            println("Cannot find MediaConvert service endpoint URL!")
            exitProcess(0)
        }
        val endpointURL = res.endpoints!!.get(0).url!!
        val mediaConvert =
            MediaConvertClient.fromEnvironment {
                region = "us-west-2"
                endpointProvider =
                    MediaConvertEndpointProvider {
                        Endpoint(endpointURL)
                    }
            }

        // output group Preset HLS low profile
        val hlsLow = createOutput("_low", "_\$dt$", 750000, 7, 1920, 1080, 640)

        // output group Preset HLS medium profile
        val hlsMedium = createOutput("_medium", "_\$dt$", 1200000, 7, 1920, 1080,
1280)

        // output group Preset HLS high profole
        val hlsHigh = createOutput("_high", "_\$dt$", 3500000, 8, 1920, 1080, 1920)

        val outputSettings =
            OutputGroupSettings {
                type = OutputGroupType.HlsGroupSettings
            }

        val outputObsList: MutableList<Output> = mutableListOf()
        if (hlsLow != null) {
            outputObsList.add(hlsLow)
        }
        if (hlsMedium != null) {
            outputObsList.add(hlsMedium)
        }
        if (hlsHigh != null) {
            outputObsList.add(hlsHigh)
        }

        // Create an OutputGroup object.
        val appleHLS =
            OutputGroup {
```

```
                name = "Apple HLS"
                customName = "Example"
                outputGroupSettings =
                    OutputGroupSettings {
                        type = OutputGroupType.HlsGroupSettings
                        this.hlsGroupSettings =
                            HlsGroupSettings {
                                directoryStructure =
HlsDirectoryStructure.SingleDirectory
                                manifestDurationFormat =
HlsManifestDurationFormat.Integer
                                streamInfResolution = HlsStreamInfResolution.Include
                                clientCache = HlsClientCache.Enabled
                                captionLanguageSetting =
HlsCaptionLanguageSetting.Omit
                                manifestCompression = HlsManifestCompression.None
                                codecSpecification = HlsCodecSpecification.Rfc4281
                                outputSelection =
HlsOutputSelection.ManifestsAndSegments
                                programDateTime = HlsProgramDateTime.Exclude
                                programDateTimePeriod = 600
                                timedMetadataId3Frame =
HlsTimedMetadataId3Frame.Priv
                                timedMetadataId3Period = 10
                                destination = fileOutput
                                segmentControl = HlsSegmentControl.SegmentedFiles
                                minFinalSegmentLength = 0.toDouble()
                                segmentLength = 4
                                minSegmentLength = 1
                            }
                    }
                outputs = outputObsList
            }

        val theOutput =
            Output {
                extension = "mp4"
                containerSettings =
                    ContainerSettings {
                        container = ContainerType.fromValue("MP4")
                    }

                videoDescription =
                    VideoDescription {
```

```
                        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 {
                                codec = VideoCodec.H264
                                h264Settings =
                                    H264Settings {
                                        rateControlMode = H264RateControlMode.Qvbr
                                        parControl =
H264ParControl.InitializeFromSource
                                        qualityTuningLevel =
H264QualityTuningLevel.SinglePass
                                        qvbrSettings = H264QvbrSettings
{ qvbrQualityLevel = 8 }
                                        codecLevel = H264CodecLevel.Auto
                                        codecProfile = H264CodecProfile.Main
                                        maxBitrate = 2400000
                                        framerateControl =
H264FramerateControl.InitializeFromSource
                                        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.DuplicateDrop
                                        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
                                }
                            }
                        }

                audioDescriptions =
                    listOf(
                        AudioDescription {
                            audioTypeControl = AudioTypeControl.FollowInput
                            languageCodeControl =
AudioLanguageCodeControl.FollowInput
                            codecSettings =
                                AudioCodecSettings {
                                    codec = AudioCodec.Aac
                                    aacSettings =
                                        AacSettings {
                                            codecProfile = AacCodecProfile.Lc
                                            rateControlMode = AacRateControlMode.Cbr
                                            codingMode = AacCodingMode.CodingMode2_0
                                            sampleRate = 44100
                                            bitrate = 160000
                                            rawFormat = AacRawFormat.None
                                            specification = AacSpecification.Mpeg4
                                            audioDescriptionBroadcasterMix =
AacAudioDescriptionBroadcasterMix.Normal
                                        }
                                }
                        },
                    )
            }
```

```
        // Create an OutputGroup
        val fileMp4 =
            OutputGroup {
                name = "File Group"
                customName = "mp4"
                outputGroupSettings =
                    OutputGroupSettings {
                        type = OutputGroupType.FileGroupSettings
                        fileGroupSettings =
                            FileGroupSettings {
                                destination = mp4Output
                            }
                    }
                outputs = listOf(theOutput)
            }

        val containerSettings1 =
            ContainerSettings {
                container = ContainerType.Raw
            }

        val thumbs =
            OutputGroup {
                name = "File Group"
                customName = "thumbs"
                outputGroupSettings =
                    OutputGroupSettings {
                        type = OutputGroupType.FileGroupSettings
                        fileGroupSettings =
                            FileGroupSettings {
                                destination = thumbsOutput
                            }
                    }

                outputs =
                    listOf(
                        Output {
                            extension = "jpg"

                            this.containerSettings = containerSettings1
                            videoDescription =
                                VideoDescription {
                                    scalingBehavior = ScalingBehavior.Default
```

```
                                    sharpness = 50
                                    antiAlias = AntiAlias.Enabled
                                    timecodeInsertion =
VideoTimecodeInsertion.Disabled
                                    colorMetadata = ColorMetadata.Insert
                                    dropFrameTimecode = DropFrameTimecode.Enabled
                                    codecSettings =
                                        VideoCodecSettings {
                                            codec = VideoCodec.FrameCapture
                                            frameCaptureSettings =
                                                FrameCaptureSettings {
                                                    framerateNumerator = 1
                                                    framerateDenominator = 1
                                                    maxCaptures = 10000000
                                                    quality = 80
                                                }
                                        }
                                }
                        },
                    )
            }

        val audioSelectors1: MutableMap<String, AudioSelector> = HashMap()
        audioSelectors1["Audio Selector 1"] =
            AudioSelector {
                defaultSelection = AudioDefaultSelection.Default
                offset = 0
            }

        val jobSettings =
            JobSettings {
                inputs =
                    listOf(
                        Input {
                            audioSelectors = audioSelectors1
                            videoSelector =
                                VideoSelector {
                                    colorSpace = ColorSpace.Follow
                                    rotate = InputRotate.Degree0
                                }
                            filterEnable = InputFilterEnable.Auto
                            filterStrength = 0
                            deblockFilter = InputDeblockFilter.Disabled
                            denoiseFilter = InputDenoiseFilter.Disabled
```

```
                            psiControl = InputPsiControl.UsePsi
                            timecodeSource = InputTimecodeSource.Embedded
                            fileInput = fileInputVal

                            outputGroups = listOf(appleHLS, thumbs, fileMp4)
                        },
                    )
            }

        val createJobRequest =
            CreateJobRequest {
                role = mcRoleARN
                settings = jobSettings
            }

        val createJobResponse = mediaConvert.createJob(createJobRequest)
        return createJobResponse.job?.id
    } catch (ex: MediaConvertException) {
        println(ex.message)
        mcClient.close()
        exitProcess(0)
    }
}

fun createOutput(
    nameModifierVal: String,
    segmentModifierVal: String,
    qvbrMaxBitrate: Int,
    qvbrQualityLevelVal: Int,
    originWidth: Int,
    originHeight: Int,
    targetWidth: Int,
): Output? {
    val targetHeight = (
        (originHeight * targetWidth / originWidth).toFloat().roundToInt() -
            (originHeight * targetWidth / originWidth).toFloat().roundToInt() % 4
        )

    var output: Output?
    try {
        val audio1 =
            AudioDescription {
                audioTypeControl = AudioTypeControl.FollowInput
                languageCodeControl = AudioLanguageCodeControl.FollowInput
```

```
                codecSettings =
                    AudioCodecSettings {
                        codec = AudioCodec.Aac
                        aacSettings =
                            AacSettings {
                                codecProfile = AacCodecProfile.Lc
                                rateControlMode = AacRateControlMode.Cbr
                                codingMode = AacCodingMode.CodingMode2_0
                                sampleRate = 44100
                                bitrate = 96000
                                rawFormat = AacRawFormat.None
                                specification = AacSpecification.Mpeg4
                                audioDescriptionBroadcasterMix =
AacAudioDescriptionBroadcasterMix.Normal
                            }
                    }
            }

        output =
            Output {
                nameModifier = nameModifierVal
                outputSettings =
                    OutputSettings {
                        hlsSettings =
                            HlsSettings {
                                segmentModifier = segmentModifierVal
                                audioGroupId = "program_audio"
                                iFrameOnlyManifest = HlsIFrameOnlyManifest.Exclude
                            }
                    }
                containerSettings =
                    ContainerSettings {
                        container = ContainerType.M3U8
                        this.m3u8Settings =
                            M3u8Settings {
                                audioFramesPerPes = 4
                                pcrControl = M3u8PcrControl.PcrEveryPesPacket
                                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 = listOf(482, 483, 484, 485, 486, 487,
488, 489, 490, 491, 492)
                            }

                        videoDescription =
                            VideoDescription {
                                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 {
                                        codec = VideoCodec.H264
                                        h264Settings =
                                            H264Settings {
                                                rateControlMode =
H264RateControlMode.Qvbr
                                                parControl =
H264ParControl.InitializeFromSource
                                                qualityTuningLevel =
H264QualityTuningLevel.SinglePass
                                                qvbrSettings =
                                                    H264QvbrSettings {
                                                        qvbrQualityLevel =
qvbrQualityLevelVal
                                                    }
                                                codecLevel = H264CodecLevel.Auto
                                                codecProfile =
                                                    if (targetHeight > 720 &&
                                                        targetWidth > 1280
                                                    ) {
                                                        H264CodecProfile.High
                                                    } else {
                                                        H264CodecProfile.Main
```

```
                                                }
                                            maxBitrate = qvbrMaxBitrate
                                            framerateControl =
H264FramerateControl.InitializeFromSource
                                            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.DuplicateDrop
                                            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
                                        }
                                    }
```

```
                                audioDescriptions = listOf(audio1)
                            }
                        }
                }
            } catch (ex: MediaConvertException) {
                println(ex.toString())
                exitProcess(0)
            }
            return output
}
```

- Para API obtener más información, consulta CreateJobla AWS SDKAPIreferencia sobre Kotlin.

`GetJob`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetJob`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getSpecificJob(
    mcClient: MediaConvertClient,
    jobId: String?,
) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }
```

```
    val endpointURL = res.endpoints!!.get(0).url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
                }
        }

    val jobRequest =
        GetJobRequest {
            id = jobId
        }

    val response: GetJobResponse = mediaConvert.getJob(jobRequest)
    println("The ARN of the job is ${response.job?.arn}.")
}
```

- Para API obtener más información, consulta GetJobla AWS SDKAPIreferencia sobre Kotlin.

`ListJobs`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListJobs`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listCompleteJobs(mcClient: MediaConvertClient) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
```

```
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }
    val endpointURL = res.endpoints!![0].url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
                }
        }

    val jobsRequest =
        ListJobsRequest {
            maxResults = 10
            status = JobStatus.fromValue("COMPLETE")
        }

    val jobsResponse = mediaConvert.listJobs(jobsRequest)
    val jobs = jobsResponse.jobs
    if (jobs != null) {
        for (job in jobs) {
            println("The JOB ARN is ${job.arn}")
        }
    }
}
```

- Para API obtener más información, consulta ListJobsla AWS SDKAPIreferencia sobre Kotlin.

Ejemplos de Amazon Pinpoint que utilizan Kotlin SDK

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- Acciones

Acciones

`CreateApp`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateApp`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequestOb =
        CreateApplicationRequest {
            name = applicationName
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequestOb
                },
            )
        return result.applicationResponse?.id
    }
}
```

- Para API obtener más información, consulta CreateAppla AWS SDKAPIreferencia sobre Kotlin.

`CreateCampaign`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateCampaign`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createPinCampaign(
    appId: String,
    segmentIdVal: String,
) {
    val scheduleOb =
        Schedule {
            startTime = "IMMEDIATE"
        }

    val defaultMessageOb =
        Message {
            action = Action.OpenApp
            body = "My message body"
            title = "My message title"
        }

    val messageConfigurationOb =
        MessageConfiguration {
            defaultMessage = defaultMessageOb
        }

    val writeCampaign =
        WriteCampaignRequest {
            description = "My description"
            schedule = scheduleOb
            name = "MyCampaign"
            segmentId = segmentIdVal
            messageConfiguration = messageConfigurationOb
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
            pinpoint.createCampaign(
                CreateCampaignRequest {
```

```
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- Para API obtener más información, consulta CreateCampaignla AWS SDKAPIreferencia sobre Kotlin.

`CreateSegment`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateSegment`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createPinpointSegment(applicationIdVal: String?): String? {
    val segmentAttributes = mutableMapOf<String, AttributeDimension>()
    val myList = mutableListOf<String>()
    myList.add("Lakers")

    val atts =
        AttributeDimension {
            attributeType = AttributeType.Inclusive
            values = myList
        }

    segmentAttributes["Team"] = atts
    val recencyDimension =
        RecencyDimension {
            duration = Duration.fromValue("DAY_30")
            recencyType = RecencyType.fromValue("ACTIVE")
        }
```

```
    val segmentBehaviors =
        SegmentBehaviors {
            recency = recencyDimension
        }

    val segmentLocation = SegmentLocation {}
    val dimensionsOb =
        SegmentDimensions {
            attributes = segmentAttributes
            behavior = segmentBehaviors
            demographic = SegmentDemographics {}
            location = segmentLocation
        }

    val writeSegmentRequestOb =
        WriteSegmentRequest {
            name = "MySegment101"
            dimensions = dimensionsOb
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse =
            pinpoint.createSegment(
                CreateSegmentRequest {
                    applicationId = applicationIdVal
                    writeSegmentRequest = writeSegmentRequestOb
                },
            )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}
```

- Para API obtener más información, consulta CreateSegmentla AWS SDKAPIreferencia sobre Kotlin.

`DeleteApp`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteApp`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deletePinApp(appId: String?) {
    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.deleteApp(
                DeleteAppRequest {
                    applicationId = appId
                },
            )
        val appName = result.applicationResponse?.name
        println("Application $appName has been deleted.")
    }
}
```

- Para API obtener más información, consulta DeleteAppla AWS SDKAPIreferencia sobre Kotlin.

`DeleteEndpoint`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteEndpoint`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deletePinEncpoint(
    appIdVal: String?,
    endpointIdVal: String?,
) {
```

```
    val deleteEndpointRequest =
        DeleteEndpointRequest {
            applicationId = appIdVal
            endpointId = endpointIdVal
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is  $id")
    }
}
```

- Para API obtener más información, consulta DeleteEndpointla AWS SDKAPIreferencia sobre Kotlin.

`GetEndpoint`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetEndpoint`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun lookupPinpointEndpoint(
    appId: String?,
    endpoint: String?,
) {
    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.getEndpoint(
                GetEndpointRequest {
                    applicationId = appId
                    endpointId = endpoint
                },
            )
```

```
        val endResponse = result.endpointResponse

        // Uses the Google Gson library to pretty print the endpoint JSON.
        val gson: com.google.gson.Gson =
            GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting()
                .create()

        val endpointJson: String = gson.toJson(endResponse)
        println(endpointJson)
    }
}
```

- Para API obtener más información, consulta GetEndpointla AWS SDKAPIreferencia sobre Kotlin.

`GetSegments`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetSegments`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listSegs(appId: String?) {
    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val response =
            pinpoint.getSegments(
                GetSegmentsRequest {
                    applicationId = appId
                },
            )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
```

```
    }
}
```

- Para API obtener más información, consulta GetSegmentsla AWS SDKAPIreferencia sobre Kotlin.

`SendMessages`

En el siguiente ejemplo de código, se muestra cómo utilizar `SendMessages`.

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val body: String =
    """
    Amazon Pinpoint test (AWS SDK for Kotlin)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK for
 Kotlin.

    """.trimIndent()

suspend fun main(args: Array<String>) {
    val 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
        toAddress - The to address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email
    """

    if (args.size != 3) {
        println(usage)
        exitProcess(0)
    }

    val subject = args[0]
    val senderAddress = args[1]
    val toAddress = args[2]
    sendEmail(subject, senderAddress, toAddress)
}

suspend fun sendEmail(
    subjectVal: String?,
    senderAddress: String,
    toAddressVal: String,
) {
    var content =
        Content {
            data = body
        }

    val messageBody =
        Body {
            text = content
        }

    val subContent =
        Content {
            data = subjectVal
        }

    val message =
        Message {
            body = messageBody
            subject = subContent
```

```
        }

    val destinationOb =
        Destination {
            toAddresses = listOf(toAddressVal)
        }

    val emailContent =
        EmailContent {
            simple = message
        }

    val sendEmailRequest =
        SendEmailRequest {
            fromEmailAddress = senderAddress
            destination = destinationOb
            this.content = emailContent
        }

    PinpointEmailClient { region = "us-east-1" }.use { pinpointemail ->
        pinpointemail.sendEmail(sendEmailRequest)
        println("Message Sent")
    }
}
```

- Para API obtener más información, consulta SendMessagesla AWS SDKAPIreferencia sobre Kotlin.

RDSEjemplos de Amazon que utilizan SDK Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con AmazonRDS.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Conceptos básicos

Conceptos básicos

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.

SDKpara Kotlin

> **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 code example, set up your development environment, including
 your credentials.
```

```
For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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

This example performs the following tasks:

1. Returns a list of the available DB engines by invoking the
 DescribeDbEngineVersions method.
2. Selects an engine family and create a custom DB parameter group by invoking the
 createDBParameterGroup method.
3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.
4. Gets parameters in the group by invoking the DescribeDbParameters method.
5. Modifies both the auto_increment_offset and auto_increment_increment parameters
 by invoking the modifyDbParameterGroup method.
6. Gets and displays the updated parameters.
7. Gets a list of allowed engine versions by invoking the describeDbEngineVersions
 method.
8. Gets a list of micro instance classes available for the selected engine.
9. Creates an Amazon Relational Database Service (Amazon RDS) database instance that
 contains a MySQL database and uses the parameter group.
10. Waits for DB instance to be ready and prints out the connection endpoint value.
11. Creates a snapshot of the DB instance.
12. Waits for the DB snapshot to be ready.
13. Deletes the DB instance.
14. Deletes the parameter group.
 */

var sleepTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>

        Where:
            dbGroupName - The database group name.
```

```
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
the database credentials.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val dbGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceIdentifier = args[2]
    val dbName = args[3]
    val dbSnapshotIdentifier = args[4]
    val secretName = args[5]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

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

    println("2. Create a custom parameter group")
    createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

    println("3. Get the parameter groups")
    describeDbParameterGroups(dbGroupName)

    println("4. Get the parameters in the group")
    describeDbParameters(dbGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBParas(dbGroupName)

    println("6. Display the updated value")
    describeDbParameters(dbGroupName, -1)
```

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

    println("8. Get a list of micro instance classes available for the selected
 engine")
    getMicroInstances()

    println("9. Create an RDS database instance that contains a MySql database and
 uses the parameter group")
    val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
 username, userPassword)
    println("The ARN of the new database is $dbARN")

    println("10. Wait for DB instance to be ready")
    waitForDbInstanceReady(dbInstanceIdentifier)

    println("11. Create a snapshot of the DB instance")
    createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("12. Wait for DB snapshot to be ready")
    waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("13. Delete the DB instance")
    deleteDbInstance(dbInstanceIdentifier)

    println("14. Delete the parameter group")
    if (dbARN != null) {
        deleteParaGroup(dbGroupName, dbARN)
    }

    println("The Scenario has successfully completed.")
}

suspend fun deleteParaGroup(
    dbGroupName: String,
    dbARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
```

```
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(dbARN) == 0) {
                        println("$dbARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database name.
                        isDataDel = true
                    }
                    index++
                }
            }
        }

        // Delete the para group.
        val parameterGroupRequest =
            DeleteDbParameterGroupRequest {
                dbParameterGroupName = dbGroupName
            }
        rdsClient.deleteDbParameterGroup(parameterGroupRequest)
        println("$dbGroupName was deleted.")
    }
}

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
```

```
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbSnapshotsRequest {
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    while (!snapshotReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbSnapshots(snapshotsRequest)
            val snapshotList: List<DbSnapshot>? = response.dbSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
```

```
) {
    val snapshotRequest =
        CreateDbSnapshotRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}

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

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

```
// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(
    dbGroupNameVal: String?,
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            dbParameterGroupName = dbGroupNameVal
            engine = "mysql"
            dbInstanceClass = "db.m4.large"
            engineVersion = "8.0"
            storageType = "standard"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

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

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "mysql"
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
```

```
            }
        }
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "mysql"
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {
                println("The engine version is ${dbEngine.engineVersion}")
                println("The engine description is ${dbEngine.dbEngineDescription}")
            }
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.Immediate
            parameterValue = "5"
        }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbParameterGroup(groupRequest)
```

```
        println("The parameter group ${response.dbParameterGroupName} was
 successfully modified")
    }
}

// Retrieve parameters in the group.
suspend fun describeDbParameters(
    dbGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
            }
        } else {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
                source = "user"
            }
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paraName: String
        if (dbParameters != null) {
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset or
 auto_increment_increment.
                paraName = para.parameterName.toString()
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    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}")
                }
            }
```

```
        }
    }
}

suspend fun describeDbParameterGroups(dbGroupName: String?) {
    val groupsRequest =
        DescribeDbParameterGroupsRequest {
            dbParameterGroupName = dbGroupName
            maxRecords = 20
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameterGroups(groupsRequest)
        val groups = response.dbParameterGroups
        if (groups != null) {
            for (group in groups) {
                println("The group name is ${group.dbParameterGroupName}")
                println("The group description is ${group.description}")
            }
        }
    }
}

// Create a parameter group.
suspend fun createDBParameterGroup(
    dbGroupName: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbParameterGroup(groupRequest)
        println("The group name is
 ${response.dbParameterGroup?.dbParameterGroupName}")
    }
}

// Returns a list of the available DB engines.
suspend fun describeDBEngines() {
    val engineVersionsRequest =
```

```
        DescribeDbEngineVersionsRequest {
            defaultOnly = true
            engine = "mysql"
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        val engines: List<DbEngineVersion>? = response.dbEngineVersions

        // Get all DbEngineVersion objects.
        if (engines != null) {
            for (engineOb in engines) {
                println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}.")
                println("The name of the database engine ${engineOb.engine}.")
                println("The version number of the database engine
 ${engineOb.engineVersion}")
            }
        }
    }
}

suspend fun getSecretValues(secretName: String?): String? {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient { region = "us-west-2" }.use { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CreateDBInstance
  - Grupo C reateDBParameter
  - CreateDBSnapshot
  - DeleteDBInstance

- eleteDBParameterGrupo D
- escribeDBEngineVersiones en D
- DescribeDBInstances
- escribeDBParameterGrupos D
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- odifyDBParameterGrupo M

Acciones

`CreateDBInstance`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateDBInstance`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNamedbVal
            engine = "mysql"
            dbInstanceClass = "db.m4.large"
            engineVersion = "8.0"
            storageType = "standard"
```

```
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

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

// Waits until the database instance is available.
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true
                    } else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
        println("Database instance is available!")
    }
}
```

- Para API obtener más información, consulta C reateDBInstance en la APIreferencia AWS SDK a Kotlin.

`DeleteDBInstance`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteDBInstance`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

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

- Para API obtener más información, consulte D eleteDBInstance en la APIreferencia AWS SDK de Kotlin.

`DescribeAccountAttributes`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeAccountAttributes`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getAccountAttributes() {
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})
        response.accountQuotas?.forEach { quotas ->
            val response = response.accountQuotas
            println("Name is: ${quotas.accountQuotaName}")
            println("Max value is ${quotas.max}")
        }
    }
}
```

- Para API obtener más información, consulta DescribeAccountAttributesla AWS SDKAPIreferencia sobre Kotlin.

`DescribeDBInstances`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeDBInstances`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeInstances() {
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest {})
        response.dbInstances?.forEach { instance ->
```

```
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")
            println("The Engine is ${instance.engine}")
            println("Connection endpoint is ${instance.endpoint?.address}")
        }
    }
}
```

- Para API obtener más información, consulte D escribeDBInstances en la APIreferencia AWS SDK de Kotlin.

`ModifyDBInstance`

En el siguiente ejemplo de código, se muestra cómo utilizar `ModifyDBInstance`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- Para API obtener más información, consulte M odifyDBInstance en la APIreferencia AWS SDK de Kotlin.

Escenarios

Crear un rastreador de elementos de trabajo de Aurora Serverless

El siguiente ejemplo de código muestra cómo crear una aplicación web que haga un seguimiento de los elementos de trabajo de una base de datos Amazon Aurora Serverless y utilice Amazon Simple Email Service (AmazonSES) para enviar informes.

SDKpara Kotlin

Muestra cómo crear una aplicación web que rastrea e informa sobre los elementos de trabajo almacenados en una RDS base de datos de Amazon.

Para obtener el código fuente completo y las instrucciones sobre cómo configurar un Spring REST API que consulte los datos de Amazon Aurora Serverless y para que lo utilice una aplicación de React, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Ejemplos RDS de Amazon Data Service que utilizan SDK Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con Amazon RDS Data Service.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- Escenarios

Escenarios

Crear un rastreador de elementos de trabajo de Aurora Serverless

El siguiente ejemplo de código muestra cómo crear una aplicación web que haga un seguimiento de los elementos de trabajo de una base de datos Amazon Aurora Serverless y utilice Amazon Simple Email Service (AmazonSES) para enviar informes.

SDKpara Kotlin

Muestra cómo crear una aplicación web que rastrea e informa sobre los elementos de trabajo almacenados en una RDS base de datos de Amazon.

Para obtener el código fuente completo y las instrucciones sobre cómo configurar un Spring REST API que consulte los datos de Amazon Aurora Serverless y para que lo utilice una aplicación de React, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Ejemplos de uso SDK de Amazon Redshift para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- Acciones
- Escenarios

Acciones

`CreateCluster`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateCluster`.

SDKpara Kotlin

> **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.

```
suspend fun createCluster(
    clusterId: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val clusterRequest =
        CreateClusterRequest {
            clusterIdentifier = clusterId
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
            nodeType = "ra3.4xlarge"
            publiclyAccessible = true
            numberOfNodes = 2
        }

    RedshiftClient { region = "us-east-1" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}
```

- Para API obtener más información, consulta CreateClusterla AWS SDKAPIreferencia sobre Kotlin.

`DeleteCluster`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteCluster`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Eliminar el clúster.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
        DeleteClusterRequest {
            clusterIdentifier = clusterId
            skipFinalClusterSnapshot = true
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- Para API obtener más información, consulta DeleteClusterla AWS SDKAPIreferencia sobre Kotlin.

`DescribeClusters`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeClusters`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Describir el clúster.

```
suspend fun describeRedshiftClusters() {
    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse =
 redshiftClient.describeClusters(DescribeClustersRequest {})
        val clusterList = clusterResponse.clusters

        if (clusterList != null) {
            for (cluster in clusterList) {
                println("Cluster database name is ${cluster.dbName}")
                println("Cluster status is ${cluster.clusterStatus}")
            }
        }
    }
}
```

- Para API obtener más información, consulta DescribeClustersla AWS SDKAPIreferencia sobre Kotlin.

`ModifyCluster`

En el siguiente ejemplo de código, se muestra cómo utilizar `ModifyCluster`.

SDKpara Kotlin

> **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

```
suspend fun modifyCluster(clusterId: String?) {
    val modifyClusterRequest =
        ModifyClusterRequest {
            clusterIdentifier = clusterId
            preferredMaintenanceWindow = "wed:07:30-wed:08:00"
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.modifyCluster(modifyClusterRequest)
        println(
            "The modified cluster was successfully modified and has
 ${clusterResponse.cluster?.preferredMaintenanceWindow} as the maintenance window",
        )
    }
}
```

- Para API obtener más información, consulta ModifyClusterla AWS SDKAPIreferencia sobre Kotlin.

Escenarios

Crear una aplicación web para realizar un seguimiento de los datos de Amazon Redshift

El siguiente ejemplo de código muestra cómo crear una aplicación web que realice el seguimiento de los elementos de trabajo y genere informes sobre ellos mediante una base de datos de Amazon Redshift.

SDKpara Kotlin

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 un Spring REST API que consulte los datos de Amazon Redshift y para que lo utilice una aplicación de React, consulte el ejemplo completo en. GitHub

Servicios utilizados en este ejemplo

- Amazon Redshift

- Amazon SES

Ejemplos de Amazon Rekognition que utilizan SDK Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Acciones

`CompareFaces`

En el siguiente ejemplo de código, se muestra cómo utilizar `CompareFaces`.

Para obtener información, consulte Comparación de rostros en imágenes.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
```

```
    sourceImageVal: String,
    targetImageVal: String,
) {
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
        Image {
            bytes = targetBytes
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches with
 ${face.confidence} % confidence.")
                }
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }
```

```
        println("Source image rotation:
 ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- Para API obtener más información, consulta CompareFaceslа AWS SDKAPIreferencia sobre Kotlin.

`CreateCollection`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateCollection`.

Para obtener información, consulte Creación de una colección.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- Para API obtener más información, consulta CreateCollectionla AWS SDKAPIreferencia sobre Kotlin.

`DeleteCollection`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteCollection`.

Para obtener información, consulte Eliminación de una colección.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- Para API obtener más información, consulta DeleteCollectionla AWS SDKAPIreferencia sobre Kotlin.

`DeleteFaces`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteFaces`.

Para obtener información, consulte Eliminación de rostros de una colección.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- Para API obtener más información, consulta DeleteFacesla AWS SDKAPIreferencia sobre Kotlin.

`DescribeCollection`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeCollection`.

Para obtener información, consulte Descripción de una colección.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- Para API obtener más información, consulta DescribeCollectionla AWS SDKAPIreferencia sobre Kotlin.

`DetectFaces`

En el siguiente ejemplo de código, se muestra cómo utilizar `DetectFaces`.

Para obtener información, consulte Detección de rostros en una imagen.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
```

```
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between ${ageRange?.low}
 and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- Para API obtener más información, consulta DetectFaceslaAWS SDKAPIreferencia sobre Kotlin.

`DetectLabels`

En el siguiente ejemplo de código, se muestra cómo utilizar `DetectLabels`.

Para obtener información, consulte Detección de etiquetas en una imagen.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }
```

```
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- Para API obtener más información, consulta DetectLabelsla AWS SDKAPIreferencia sobre Kotlin.

`DetectModerationLabels`

En el siguiente ejemplo de código, se muestra cómo utilizar `DetectModerationLabels`.

Para obtener información, consulte Detección de imágenes inapropiadas.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
```

```
            println("Label: ${label.name} - Confidence: ${label.confidence} %
 Parent: ${label.parentName}")
        }
    }
}
```

- Para API obtener más información, consulta DetectModerationLabelsla AWS SDKAPIreferencia sobre Kotlin.

`DetectText`

En el siguiente ejemplo de código, se muestra cómo utilizar `DetectText`.

Para obtener información, consulte Detección de texto en una imagen.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
```

```
            println("Type: ${text.type}")
        }
    }
}
```

- Para API obtener más información, consulta DetectTextla AWS SDKAPIreferencia sobre Kotlin.

`IndexFaces`

En el siguiente ejemplo de código, se muestra cómo utilizar `IndexFaces`.

Para obtener información, consulte Adición de rostros a una colección.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)
```

```
        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")

            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

- Para API obtener más información, consulta IndexFaceslа AWS SDKAPIreferencia sobre Kotlin.

`ListCollections`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListCollections`.

Para obtener información, consulte Enumerar colecciones.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
```

```
            maxResults = 10
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- Para API obtener más información, consulta ListCollectionsla AWS SDKAPIreferencia sobre Kotlin.

`ListFaces`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListFaces`.

Para obtener información, consulte Enumerar rostros en una colección.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
```

```
    }
}
```

- Para API obtener más información, consulta ListFaces la AWS SDKAPIreferencia sobre Kotlin.

`RecognizeCelebrities`

En el siguiente ejemplo de código, se muestra cómo utilizar `RecognizeCelebrities`.

Para obtener información, consulte Reconocimiento de famosos en una imagen.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
```

```
    }
}
```

- Para API obtener más información, consulta RecognizeCelebritiesla AWS SDKAPIreferencia sobre Kotlin.

Escenarios

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDKpara Kotlin

Muestra cómo desarrollar una aplicación de administración 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

- Gateway de API
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detectar información en vídeos

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.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Detecte rostros en un vídeo almacenado en un bucket de Amazon S3.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidOb
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}
```

```
suspend fun getFaceResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest =
            GetFaceDetectionRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
            status = response.jobStatus.toString()
            if (status.compareTo("SUCCEEDED") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}
```

Detecte contenido inapropiado u ofensivo en un vídeo almacenado en un bucket de Amazon S3.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidOb
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest =
            GetContentModerationRequest {
                jobId = startJobId
                maxResults = 10
            }
```

```
        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("SUCCEEDED") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = modDetectionResponse?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        modDetectionResponse?.moderationLabels?.forEach { mod ->
            val seconds: Long = mod.timestamp / 1000
            print("Mod label: $seconds ")
            println(mod.moderationLabel)
        }
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration

- StartLabelDetection
- StartPersonTracking
- StartSegmentDetection
- StartTextDetection

Detectar objetos en imágenes

El siguiente ejemplo de código muestra cómo crear una aplicación que utilice Amazon Rekognition para detectar objetos por categoría en las imágenes.

SDKpara Kotlin

Muestra cómo utilizar Amazon API Rekognition Kotlin 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 (AmazonSES).

Para obtener el código fuente completo y las instrucciones sobre cómo configurarla y ejecutarla, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Ejemplos de registro de dominios de Route 53 utilizando SDK Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el registro de dominios AWS SDK para Kotlin con Route 53.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el 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.

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <domainType>

       Where:
           domainType - The domain type (for example, com).
    """

    if (args.size != 1) {
        println(usage)
```

```
        exitProcess(0)
    }

    val domainType = args[0]
    println("Invokes ListPrices using a Paginated method.")
    listPricesPaginated(domainType)
}

suspend fun listPricesPaginated(domainType: String) {
    val pricesRequest =
        ListPricesRequest {
            maxItems = 10
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- Para API obtener más información, consulta ListPricesla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Conceptos básicos
- Acciones

Conceptos básicos

Conceptos básicos

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.

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin 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.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <domainType> <phoneNumber> <email> <domainSuggestion> <firstName>
 <lastName> <city>
        Where:
           domainType - The domain type (for example, com).
           phoneNumber - The phone number to use (for example, +1.2065550100)
           email - The email address to use.
           domainSuggestion - The domain suggestion (for example, findmy.example).
           firstName - The first name to use to register a domain.
           lastName -  The last name to use to register a domain.
           city - The city to use to register a domain.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val domainType = args[0]
    val phoneNumber = args[1]
    val email = args[2]
    val domainSuggestion = args[3]
    val firstName = args[4]
    val lastName = args[5]
    val city = args[6]

    println(DASHES)
    println("Welcome to the Amazon Route 53 domains example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List current domains.")
    listDomains()
    println(DASHES)
```

```
    println(DASHES)
    println("2. List operations in the past year.")
    listOperations()
    println(DASHES)

    println(DASHES)
    println("3. View billing for the account in the past year.")
    listBillingRecords()
    println(DASHES)

    println(DASHES)
    println("4. View prices for domain types.")
    listAllPrices(domainType)
    println(DASHES)

    println(DASHES)
    println("5. Get domain suggestions.")
    listDomainSuggestions(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("6. Check domain availability.")
    checkDomainAvailability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("7. Check domain transferability.")
    checkDomainTransferability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("8. Request a domain registration.")
    val opId = requestDomainRegistration(domainSuggestion, phoneNumber, email,
firstName, lastName, city)
    println(DASHES)

    println(DASHES)
    println("9. Get operation details.")
    getOperationalDetail(opId)
    println(DASHES)

    println(DASHES)
    println("10. Get domain details.")
    println("Note: You must have a registered domain to get details.")
```

```
    println("Otherwise an exception is thrown that states ")
    println("Domain xxxxxxx not found in xxxxxxx account.")
    getDomainDetails(domainSuggestion)
    println(DASHES)
}

suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}

suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}

suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
```

```
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}

suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}

suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
```

```
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}

suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}

suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
```

```
            }
    }
}

suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}

suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
```

```
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}

suspend fun listDomains() {
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Acciones

`CheckDomainAvailability`

En el siguiente ejemplo de código, se muestra cómo utilizar `CheckDomainAvailability`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}
```

- Para API obtener más información, consulta CheckDomainAvailabilityla AWS SDKAPIreferencia sobre Kotlin.

`CheckDomainTransferability`

En el siguiente ejemplo de código, se muestra cómo utilizar `CheckDomainTransferability`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}
```

- Para API obtener más información, consulta CheckDomainTransferabilityla AWS SDKAPIreferencia sobre Kotlin.

`GetDomainDetail`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetDomainDetail`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
```

```
}
```

- Para API obtener más información, consulta GetDomainDetailla AWS SDKAPIreferencia sobre Kotlin.

`GetDomainSuggestions`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetDomainSuggestions`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}
```

- Para API obtener más información, consulta GetDomainSuggestionsla AWS SDKAPIreferencia sobre Kotlin.

`GetOperationDetail`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetOperationDetail`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

- Para API obtener más información, consulta GetOperationDetailla AWS SDKAPIreferencia sobre Kotlin.

`ListDomains`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListDomains`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listDomains() {
```

```
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- Para API obtener más información, consulta ListDomainsla AWS SDKAPIreferencia sobre Kotlin.

`ListOperations`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListOperations`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
```

```
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}
```

- Para API obtener más información, consulta ListOperationsla AWS SDKAPIreferencia sobre Kotlin.

`ListPrices`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListPrices`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
```

```
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- Para API obtener más información, consulta ListPricesla AWS SDKAPIreferencia sobre Kotlin.

`RegisterDomain`

En el siguiente ejemplo de código, se muestra cómo utilizar `RegisterDomain`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
```

```
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}
```

- Para API obtener más información, consulta RegisterDomainla AWS SDKAPIreferencia sobre Kotlin.

`ViewBilling`

En el siguiente ejemplo de código, se muestra cómo utilizar `ViewBilling`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
```

```
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}
```

- Para API obtener más información, consulta ViewBillingla AWS SDKAPIreferencia sobre Kotlin.

Ejemplos de Amazon S3 que utilizan SDK Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con Amazon S3.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Conceptos básicos

Conceptos básicos

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.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun main(args: Array<String>) {
    val 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).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)

    // Download the object to another local file.
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
```

```
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
```

```
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
```

```
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Acciones

`CopyObject`

En el siguiente ejemplo de código, se muestra cómo utilizar `CopyObject`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
```

```
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- Para API obtener más información, consulta CopyObjectla AWS SDKAPIreferencia sobre Kotlin.

`CreateBucket`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateBucket`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- Para API obtener más información, consulta CreateBucketla AWS SDKAPIreferencia sobre Kotlin.

`CreateMultiRegionAccessPoint`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateMultiRegionAccessPoint`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Configure el cliente de control de S3 para enviar la solicitud a la región us-west-2.

```
        suspend fun createS3ControlClient(): S3ControlClient {
            // Configure your S3ControlClient to send requests to US West (Oregon).
            val s3Control = S3ControlClient.fromEnvironment {
                region = "us-west-2"
            }
            return s3Control
        }
```

Cree el punto de acceso de varias regiones.

```
    suspend fun createMrap(
        s3Control: S3ControlClient,
        accountIdParam: String,
        bucketName1: String,
        bucketName2: String,
        mrapName: String,
    ): String {
        println("Creating MRAP ...")
        val createMrapResponse: CreateMultiRegionAccessPointResponse =
            s3Control.createMultiRegionAccessPoint {
                accountId = accountIdParam
                clientToken = UUID.randomUUID().toString()
                details {
                    name = mrapName
                    regions = listOf(
                        Region {
                            bucket = bucketName1
```

```
                        },
                        Region {
                            bucket = bucketName2
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn

        // Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

Espere a que el punto de acceso de varias regiones esté disponible.

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )
```

```
            var status: String? = describeResponse.asyncOperation?.requestStatus
            while (status != "SUCCEEDED") {
                delay(timeBetweenChecks)
                describeResponse =
 s3Control.describeMultiRegionAccessPointOperation(
                    input = DescribeMultiRegionAccessPointOperationRequest {
                        accountId = accountIdParam
                        requestTokenArn = requestToken
                    },
                )
                status = describeResponse.asyncOperation?.requestStatus
                println(status)
            }
        }
```

- Para obtener más información, consulta la guía AWS SDK para desarrolladores de Kotlin.
- Para API obtener más información, consulta CreateMultiRegionAccessPointla referencia AWS SDK sobre Kotlin API.

`DeleteBucketPolicy`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteBucketPolicy`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
```

```
        println("Done!")
    }
}
```

- Para API obtener más información, consulta DeleteBucketPolicyla AWS SDKAPIreferencia sobre Kotlin.

`DeleteObjects`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteObjects`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
```

```
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- Para API obtener más información, consulta DeleteObjectsla AWS SDKAPIreferencia sobre Kotlin.

`GetBucketPolicy`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetBucketPolicy`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- Para API obtener más información, consulta GetBucketPolicyla AWS SDKAPIreferencia sobre Kotlin.

`GetObject`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObject`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- Para API obtener más información, consulta GetObjectla AWS SDKAPIreferencia sobre Kotlin.

`GetObjectAcl`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObjectAcl`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- Para API obtener más información, consulta GetObjectAcllla AWS SDKAPIreferencia sobre Kotlin.

`ListObjectsV2`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListObjectsV2`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- Para API obtener más información, consulta la ListObjectsversión 2 en la sección AWS SDKde API referencia sobre Kotlin.

`PutBucketAcl`

En el siguiente ejemplo de código, se muestra cómo utilizar `PutBucketAcl`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
```

```
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- Para API obtener más información, consulta PutBucketAclla AWS SDKAPIreferencia sobre Kotlin.

`PutObject`

En el siguiente ejemplo de código, se muestra cómo utilizar `PutObject`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- Para API obtener más información, consulta PutObjectla AWS SDKAPIreferencia sobre Kotlin.

Escenarios

Crea un prefirmado URL

El siguiente ejemplo de código muestra cómo crear un objeto prefirmado URL para Amazon S3 y cómo cargar un objeto.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Crea una solicitud `GetObject` prefirmada y úsala URL para descargar un objeto.

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET request
 to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

Cree una solicitud prefirmada `GetObject` con opciones avanzadas.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
```

```
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be used
 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

Cree una solicitud prefirmada de `PutObject` y úsela para subir un objeto.

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
 HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
```

```
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }.put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- Para obtener más información, consulta la guía AWS SDKpara desarrolladores de Kotlin.

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDKpara Kotlin

Muestra cómo desarrollar una aplicación de administración 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

- Gateway de API
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detectar objetos en imágenes

El siguiente ejemplo de código muestra cómo crear una aplicación que utilice Amazon Rekognition para detectar objetos por categoría en las imágenes.

SDKpara Kotlin

Muestra cómo utilizar Amazon API Rekognition Kotlin 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 (AmazonSES).

Para obtener el código fuente completo y las instrucciones sobre cómo configurarla y ejecutarla, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Obtención de un objeto desde un punto de acceso de varias regiones

En el siguiente ejemplo de código se muestra cómo obtener un objeto desde un punto de acceso de varias regiones.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Configure el cliente S3 para que utilice el algoritmo de firma asimétrica Sigv4 (SigV4a).

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric Sigv4 (Sigv4a) signing
algorithm.
```

```
            val sigV4AScheme = SigV4AsymmetricAuthScheme(CrtAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4AScheme)
            }
            return s3
        }
```

Usa el punto de acceso multirregional ARN en lugar del nombre de un depósito para recuperar el objeto.

```
    suspend fun getObjectFromMrap(
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
            key = keyName
        }

        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
        return stringObj
    }
```

- Para obtener más información, consulta la guía AWS SDKpara desarrolladores de Kotlin.
- Para API obtener más información, consulta GetObjectla referencia AWS SDK sobre Kotlin API.

SageMaker Ejemplos de IA que se utilizan SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con SageMaker IA.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Introducción

Hola, AI SageMaker

Los siguientes ejemplos de código muestran cómo empezar a utilizar la SageMaker IA.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listBooks() {
    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.listNotebookInstances(ListNotebookInstancesRequest {})
        response.notebookInstances?.forEach { item ->
            println("The notebook name is: ${item.notebookInstanceName}")
        }
    }
}
```

- Para API obtener más información, consulta ListNotebookInstancesla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Acciones

- Escenarios

Acciones

`CreatePipeline`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreatePipeline`.

SDKpara Kotlin

> **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 a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
```

```
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}
```

- Para API obtener más información, consulta CreatePipelinela AWS SDKAPIreferencia sobre Kotlin.

`DeletePipeline`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeletePipeline`.

SDKpara Kotlin

> **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 a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}
```

- Para API obtener más información, consulta DeletePipelinela AWS SDKAPIreferencia sobre Kotlin.

`DescribePipelineExecution`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribePipelineExecution`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}
```

- Para API obtener más información, consulta DescribePipelineExecutionla AWS SDKAPIreferencia sobre Kotlin.

`StartPipelineExecution`

En el siguiente ejemplo de código, se muestra cómo utilizar `StartPipelineExecution`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
 pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
```

```
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
```

```
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}
```

- Para API obtener más información, consulta StartPipelineExecutionla AWS SDKAPIreferencia sobre Kotlin.

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, consulta Cómo crear y ejecutar SageMaker canalizaciones con AWS SDKs Community.aws.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
val DASHES = String(CharArray(80)).replace("\u0000", "-")
private var eventSourceMapping = ""

suspend fun main(args: Array<String>) {
```

```
    val usage = """
    Usage:
          <sageMakerRoleName> <lambdaRoleName> <functionName> <functionKey>
 <queueName> <bucketName> <bucketFunction> <lnglatData> <spatialPipelinePath>
 <pipelineName>

    Where:
        sageMakerRoleName - The name of the Amazon SageMaker role.
        lambdaRoleName - The name of the AWS Lambda role.
        functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).
        functionKey - The name of the Amazon S3 key name that represents the Lambda
 function (for example, SageMakerLambda.zip).
        queueName - The name of the Amazon Simple Queue Service (Amazon SQS) queue.
        bucketName - The name of the Amazon Simple Storage Service (Amazon S3)
 bucket.
        bucketFunction - The name of the Amazon S3 bucket that contains the Lambda
 ZIP file.
        lnglatData - The file location of the latlongtest.csv file required for this
 use case.
        spatialPipelinePath - The file location of the GeoSpatialPipeline.json file
 required for this use case.
        pipelineName - The name of the pipeline to create (for example, sagemaker-
sdk-example-pipeline).
    """

    if (args.size != 10) {
        println(usage)
        exitProcess(1)
    }

    val sageMakerRoleName = args[0]
    val lambdaRoleName = args[1]
    val functionKey = args[2]
    val functionName = args[3]
    val queueName = args[4]
    val bucketName = args[5]
    val bucketFunction = args[6]
    val lnglatData = args[7]
    val spatialPipelinePath = args[8]
    val pipelineName = args[9]
    val handlerName = "org.example.SageMakerLambdaFunction::handleRequest"

    println(DASHES)
```

```
    println("Welcome to the Amazon SageMaker pipeline example scenario.")
    println(
        """
         This example workflow will guide you through setting up and running an
         Amazon SageMaker pipeline. The pipeline uses an AWS Lambda function and an
         Amazon SQS Queue. It runs a vector enrichment reverse geocode job to
         reverse geocode addresses in an input file and store the results in an
export file.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("First, we will set up the roles, functions, and queue needed by the
SageMaker pipeline.")
    val lambdaRoleArn: String = checkLambdaRole(lambdaRoleName)
    val sageMakerRoleArn: String = checkSageMakerRole(sageMakerRoleName)
    val functionArn = checkFunction(functionName, bucketFunction, functionKey,
handlerName, lambdaRoleArn)
    val queueUrl = checkQueue(queueName, functionName)
    println(DASHES)

    println(DASHES)
    println("Setting up bucket $bucketName")
    if (!checkBucket(bucketName)) {
        setupBucket(bucketName)
        println("Put $lnglatData into $bucketName")
        val objectKey = "samplefiles/latlongtest.csv"
        putS3Object(bucketName, objectKey, lnglatData)
    }
    println(DASHES)

    println(DASHES)
    println("Now we can create and run our pipeline.")
    setupPipeline(spatialPipelinePath, sageMakerRoleArn, functionArn, pipelineName)
    val pipelineExecutionARN = executePipeline(bucketName, queueUrl,
sageMakerRoleArn, pipelineName)
    println("The pipeline execution ARN value is $pipelineExecutionARN")
    waitForPipelineExecution(pipelineExecutionARN)
    println("Wait 30 secs to get output results $bucketName")
    TimeUnit.SECONDS.sleep(30)
    getOutputResults(bucketName)
    println(DASHES)
```

```
    println(DASHES)
    println(
        """
            The pipeline has completed. To view the pipeline and runs in SageMaker
 Studio, follow these instructions:
            https://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-studio.html
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("Do you want to delete the AWS resources used in this Workflow? (y/n)")
    val `in` = Scanner(System.`in`)
    val delResources = `in`.nextLine()
    if (delResources.compareTo("y") == 0) {
        println("Lets clean up the AWS resources. Wait 30 seconds")
        TimeUnit.SECONDS.sleep(30)
        deleteEventSourceMapping(functionName)
        deleteSQSQueue(queueName)
        listBucketObjects(bucketName)
        deleteBucket(bucketName)
        delLambdaFunction(functionName)
        deleteLambdaRole(lambdaRoleName)
        deleteSagemakerRole(sageMakerRoleName)
        deletePipeline(pipelineName)
    } else {
        println("The AWS Resources were not deleted!")
    }
    println(DASHES)

    println(DASHES)
    println("SageMaker pipeline scenario is complete.")
    println(DASHES)
}

// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
```

```
    }
}

suspend fun deleteSagemakerRole(roleNameVal: String) {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in sageMakerRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteLambdaRole(roleNameVal: String) {
    val lambdaRolePolicies = getLambdaRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in lambdaRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}
```

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}

suspend fun deleteBucketObjects(bucketName: String, objectName: String?) {
    val toDelete = ArrayList<ObjectIdentifier>()
    val obId = ObjectIdentifier {
        key = objectName
    }
    toDelete.add(obId)
    val delOb = Delete {
        objects = toDelete
    }
    val dor = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.deleteObjects(dor)
        println("*** $bucketName objects were deleted.")
    }
}

suspend fun listBucketObjects(bucketNameVal: String) {
    val listObjects = ListObjectsRequest {
```

```
        bucket = bucketNameVal
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val objects = res.contents
        if (objects != null) {
            for (myValue in objects) {
                println("The name of the key is ${myValue.key}")
                deleteBucketObjects(bucketNameVal, myValue.key)
            }
        }
    }
}

// Delete the specific Amazon SQS queue.
suspend fun deleteSQSQueue(queueNameVal: String?) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val urlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = urlVal
        }
        sqsClient.deleteQueue(deleteQueueRequest)
    }
}

// Delete the queue event mapping.
suspend fun deleteEventSourceMapping(functionNameVal: String) {
    if (eventSourceMapping.compareTo("") == 0) {
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val request = ListEventSourceMappingsRequest {
                functionName = functionNameVal
            }
            val response = lambdaClient.listEventSourceMappings(request)
            val eventList = response.eventSourceMappings
            if (eventList != null) {
                for (event in eventList) {
                    eventSourceMapping = event.uuid.toString()
                }
            }
```

```
        }
    }

    val eventSourceMappingRequest = DeleteEventSourceMappingRequest {
        uuid = eventSourceMapping
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest)
        println("The event mapping is deleted!")
    }
}

// Reads the objects in the S3 bucket and displays the values.
private suspend fun readObject(bucketName: String, keyVal: String?) {
    println("Output file contents: \n")
    val objectRequest = GetObjectRequest {
        bucket = bucketName
        key = keyVal
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.getObject(objectRequest) { resp ->
            val byteArray = resp.body?.toByteArray()
            val text = byteArray?.let { String(it, StandardCharsets.UTF_8) }
            println("Text output: $text")
        }
    }
}

// Display the results from the output directory.
suspend fun getOutputResults(bucketName: String?) {
    println("Getting output results $bucketName.")
    val listObjectsRequest = ListObjectsRequest {
        bucket = bucketName
        prefix = "outputfiles/"
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val response = s3Client.listObjects(listObjectsRequest)
        val s3Objects: List<Object>? = response.contents
        if (s3Objects != null) {
            for (`object` in s3Objects) {
                if (bucketName != null) {
                    readObject(bucketName, (`object`.key))
                }
            }
```

```
        }
    }
}

suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}

// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
 pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
```

```
        name = "parameter_queue_url"
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }
```

```
    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}

// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
```

```
            pipelineDefinition = jsonObject.toString()
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}

suspend fun putS3Object(bucketName: String, objectKey: String, objectPath: String) {
    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        body = File(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putObject(request)
        println("Successfully placed $objectKey into bucket $bucketName")
    }
}

suspend fun setupBucket(bucketName: String) {
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String): Boolean {
    try {
        val headBucketRequest = HeadBucketRequest {
            bucket = bucketName
        }
        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            println("$bucketName exists")
            return true
        }
    } catch (e: S3Exception) {
```

```
        println("Bucket does not exist")
    }
    return false
}

// Connect the queue to the Lambda function as an event source.
suspend fun connectLambda(queueUrlVal: String?, lambdaNameVal: String?) {
    println("Connecting the Lambda function and queue for the pipeline.")
    var queueArn = ""

    // Specify the attributes to retrieve.
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)
    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val queueAtts = response.attributes
        if (queueAtts != null) {
            for ((key, value) in queueAtts) {
                println("Key = $key, Value = $value")
                queueArn = value
            }
        }
    }
    val eventSourceMappingRequest = CreateEventSourceMappingRequest {
        eventSourceArn = queueArn
        functionName = lambdaNameVal
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        val response1 =
 lambdaClient.createEventSourceMapping(eventSourceMappingRequest)
        eventSourceMapping = response1.uuid.toString()
        println("The mapping between the event source and Lambda function was
 successful")
    }
}

// Set up the SQS queue to use with the pipeline.
suspend fun setupQueue(queueNameVal: String, lambdaNameVal: String): String {
    println("Setting up queue named $queueNameVal")
```

```
    val queueAtt: MutableMap<String, String> = HashMap()
    queueAtt.put("DelaySeconds", "5")
    queueAtt.put("ReceiveMessageWaitTimeSeconds", "5")
    queueAtt.put("VisibilityTimeout", "300")

    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
        attributes = queueAtt
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("\nGet queue url")
        val getQueueUrlResponse = sqsClient.getQueueUrl(GetQueueUrlRequest
 { queueName = queueNameVal })
        TimeUnit.SECONDS.sleep(15)
        connectLambda(getQueueUrlResponse.queueUrl, lambdaNameVal)
        println("Queue ready with Url " + getQueueUrlResponse.queueUrl)
        return getQueueUrlResponse.queueUrl.toString()
    }
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a new
 queue
// and returns the ARN value.
suspend fun checkQueue(queueNameVal: String, lambdaNameVal: String): String? {
    println("Checking to see if the queue exists. If not, a new queue will be
 created for use in this workflow.")
    var queueUrl: String
    try {
        val request = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-west-2" }.use { sqsClient ->
            val response = sqsClient.getQueueUrl(request)
            queueUrl = response.queueUrl.toString()
            println(queueUrl)
        }
    } catch (e: SqsException) {
        println(e.message + " A new queue will be created")
        queueUrl = setupQueue(queueNameVal, lambdaNameVal)
    }
    return queueUrl
```

```
}

suspend fun createNewFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java11
        memorySize = 1024
        timeout = 200
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        println("${functionResponse.functionArn} was created")
        return functionResponse.functionArn.toString()
    }
}

suspend fun checkFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    println("Checking to see if the function exists. If not, a new AWS Lambda
 function will be created for use in this workflow.")
    var functionArn: String
    try {
        // Does this function already exist.
        val functionRequest = GetFunctionRequest {
            functionName = myFunctionName
        }
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val response = lambdaClient.getFunction(functionRequest)
            functionArn = response.configuration?.functionArn.toString()
            println("$functionArn exists")
```

```
        }
    } catch (e: LambdaException) {
        println(e.message + " A new function will be created")
        functionArn = createNewFunction(myFunctionName, s3BucketName, myS3Key,
 myHandler, myRole)
    }
    return functionArn
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
suspend fun checkSageMakerRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS SageMaker to use.")
    var roleArn: String
    try {
        val roleRequest = GetRoleRequest {
            roleName = roleNameVal
        }
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createSageMakerRole(roleNameVal)
    }
    return roleArn
}

suspend fun createSageMakerRole(roleNameVal: String): String {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    println("Creating a role to use with SageMaker.")
    val 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\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in sageMakerRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.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.toString()
    }
}

// Checks to see if the Lambda role exists. If not, this method creates it.
suspend fun checkLambdaRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS Lambda to use.")
    var roleArn: String
    val roleRequest = GetRoleRequest {
        roleName = roleNameVal
    }

    try {
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
```

```
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createLambdaRole(roleNameVal)
    }

    return roleArn
}

private suspend fun createLambdaRole(roleNameVal: String): String {
    val lambdaRolePolicies = getLambdaRolePolicies()
    val 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\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in lambdaRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }
```

```
        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
        println("Role ready with ARN " + roleResult.role?.arn)
        return roleResult.role?.arn.toString()
    }
}

fun getLambdaRolePolicies(): Array<String?> {
    val lambdaRolePolicies = arrayOfNulls<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
}

fun getSageMakerRolePolicies(): Array<String?> {
    val sageMakerRolePolicies = arrayOfNulls<String>(3)
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    return sageMakerRolePolicies
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Ejemplos de Secrets Manager que se utilizan SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin with 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Acciones

`GetSecretValue`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetSecretValue`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
```

```
    }
}
```

- Para API obtener más información, consulta GetSecretValuela AWS SDKAPIreferencia sobre Kotlin.

SESEjemplos de Amazon que utilizan SDK Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con AmazonSES.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- Escenarios

Escenarios

Creación de una aplicación web para hacer un seguimiento de los datos de DynamoDB

El siguiente ejemplo de código muestra cómo crear una aplicación web que haga un seguimiento de los elementos de trabajo de una tabla de Amazon DynamoDB y utilice Amazon Simple Email Service (SESAmazon) para enviar informes.

SDKpara Kotlin

Muestra cómo utilizar Amazon DynamoDB para crear una aplicación web API dinámica que realice un seguimiento de los datos de trabajo de DynamoDB.

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 SES

Crear una aplicación web para realizar un seguimiento de los datos de Amazon Redshift

El siguiente ejemplo de código muestra cómo crear una aplicación web que realice el seguimiento de los elementos de trabajo y genere informes sobre ellos mediante una base de datos de Amazon Redshift.

SDKpara Kotlin

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 un Spring REST API que consulte los datos de Amazon Redshift y para que lo 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

El siguiente ejemplo de código muestra cómo crear una aplicación web que haga un seguimiento de los elementos de trabajo de una base de datos Amazon Aurora Serverless y utilice Amazon Simple Email Service (AmazonSES) para enviar informes.

SDKpara Kotlin

Muestra cómo crear una aplicación web que rastrea e informa sobre los elementos de trabajo almacenados en una RDS base de datos de Amazon.

Para obtener el código fuente completo y las instrucciones sobre cómo configurar un Spring REST API que consulte los datos de Amazon Aurora Serverless y para que lo utilice una aplicación de React, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Aurora

- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Detectar objetos en imágenes

El siguiente ejemplo de código muestra cómo crear una aplicación que utilice Amazon Rekognition para detectar objetos por categoría en las imágenes.

SDKpara Kotlin

Muestra cómo utilizar Amazon API Rekognition Kotlin 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 (AmazonSES).

Para obtener el código fuente completo y las instrucciones sobre cómo configurarla y ejecutarla, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

SNSEjemplos de Amazon que utilizan SDK Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con AmazonSNS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Introducción

Hola Amazon SNS

Los siguientes ejemplos de código muestran cómo empezar a utilizar AmazonSNS.

SDKpara Kotlin

> **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 aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
```

```
}
```

- Para API obtener más información, consulta ListTopicsla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Acciones
- Escenarios

Acciones

`CreateTopic`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateTopic`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- Para API obtener más información, consulta CreateTopicla AWS SDKAPIreferencia sobre Kotlin.

`DeleteTopic`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteTopic`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- Para API obtener más información, consulta DeleteTopicla AWS SDKAPIreferencia sobre Kotlin.

`GetTopicAttributes`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetTopicAttributes`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- Para API obtener más información, consulta GetTopicAttributesla AWS SDKAPIreferencia sobre Kotlin.

`ListSubscriptions`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListSubscriptions`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listSNSSubscriptions() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- Para API obtener más información, consulta ListSubscriptionsla AWS SDKAPIreferencia sobre Kotlin.

`ListTopics`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListTopics`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listSNSTopics() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- Para API obtener más información, consulta ListTopicsla AWS SDKAPIreferencia sobre Kotlin.

`Publish`

En el siguiente ejemplo de código, se muestra cómo utilizar `Publish`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Para API obtener más información, consulta Publicar en AWS SDK como APIreferencia sobre Kotlin.

`SetTopicAttributes`

En el siguiente ejemplo de código, se muestra cómo utilizar `SetTopicAttributes`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
```

```
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- Para API obtener más información, consulta SetTopicAttributesla AWS SDKAPIreferencia sobre Kotlin.

`Subscribe`

En el siguiente ejemplo de código, se muestra cómo utilizar `Subscribe`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscriba una dirección de correo electrónico a un tema.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
```

```
    }
}
```

Suscriba una función de Lambda a un tema.

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Para API obtener más información, consulta Suscríbete AWS SDKpara obtener información sobre Kotlin API.

`TagResource`

En el siguiente ejemplo de código, se muestra cómo utilizar `TagResource`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun addTopicTags(topicArn: String) {
```

```
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

    val tag2 =
        Tag {
            key = "Environment"
            value = "Gamma"
        }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request =
        TagResourceRequest {
            resourceArn = topicArn
            tags = tagList
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- Para API obtener más información, consulta TagResourcela AWS SDKAPIreferencia sobre Kotlin.

`Unsubscribe`

En el siguiente ejemplo de código, se muestra cómo utilizar `Unsubscribe`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Para API obtener más información, consulta Darse de baja como APIreferencia sobre Kotlin.AWS SDK

Escenarios

Creación de una aplicación para enviar datos a una tabla de DynamoDB

El siguiente ejemplo de código muestra cómo crear una aplicación que envíe datos a una tabla de Amazon DynamoDB y le notifique cuando un usuario actualice la tabla.

SDKpara Kotlin

Muestra cómo crear una aplicación nativa de Android que envíe datos mediante Amazon API DynamoDB Kotlin y envíe un mensaje de texto mediante Amazon Kotlin. SNS API

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 SNS

Creación de una SNS aplicación de Amazon

El siguiente ejemplo de código muestra cómo crear una aplicación que tenga funciones de suscripción y publicación y que traduzca los mensajes.

SDKpara Kotlin

Muestra cómo usar Amazon SNS Kotlin API para crear una aplicación que tenga funciones de suscripción y publicación. Además, esta aplicación de ejemplo también traduce los mensajes.

Para obtener el código fuente completo y las instrucciones sobre cómo crear una aplicación web, consulte el ejemplo completo en GitHub.

Para obtener el código fuente completo y las instrucciones sobre cómo crear una aplicación Android nativa, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon SNS
- Amazon Translate

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDKpara Kotlin

Muestra cómo desarrollar una aplicación de administración 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

- Gateway de API
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Publica un mensaje SMS de texto

El siguiente ejemplo de código muestra cómo publicar SMS mensajes con AmazonSNS.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Para API obtener más información, consulta Publicar en AWS SDK como APIreferencia sobre Kotlin.

Publicación de mensajes en colas

En el siguiente ejemplo de código, se muestra cómo:

- Crea un tema (FIFOo noFIFO).
- Suscribir 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

SDKpara Kotlin

> **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 aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
```

```
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following 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.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
```

```
    println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
    println(
        """
                In this workflow, you will create an SNS topic and subscribe an SQS
 queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication and
 message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically generated
 from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://docs.aws.amazon.com/
sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
```

```
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended to
the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
```

```
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/latest/
dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's subscription
 to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
```

```
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following \"tone
\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
 deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
```

```
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
 continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
```

```
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }
```

```
    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
```

```
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
```

```
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
```

```
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic " +
 topicArnVal + "\n" +
                     "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}
```

```
suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
```

```
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}
```

```
suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

SQSEjemplos de Amazon que utilizan SDK Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con AmazonSQS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Introducción

Hola Amazon SQS

Los siguientes ejemplos de código muestran cómo empezar a utilizar AmazonSQS.

SDKpara Kotlin

> **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.kotlin.sqs

import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.paginators.listQueuesPaginated
import kotlinx.coroutines.flow.transform

suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SqsClient { region = "us-east-1" }.use { sqsClient ->
```

```
        sqsClient
            .listQueuesPaginated { }
            .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
            .collect { queue ->
                println("The Queue URL is $queue")
            }
    }
}
```

- Para API obtener más información, consulta ListQueuesla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Acciones
- Escenarios

Acciones

`CreateQueue`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateQueue`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
```

```
        println("Get queue url")

        val getQueueUrlRequest =
            GetQueueUrlRequest {
                queueName = queueNameVal
            }

        val getQueueUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
        return getQueueUrlResponse.queueUrl.toString()
    }
}
```

- Para API obtener más información, consulta CreateQueuela AWS SDKAPIreferencia sobre Kotlin.

`DeleteMessage`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteMessage`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}
```

```
suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Para API obtener más información, consulta DeleteMessagela AWS SDKAPIreferencia sobre Kotlin.

`DeleteQueue`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteQueue`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
```

```
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Para API obtener más información, consulta DeleteQueuela AWS SDKAPIreferencia sobre Kotlin.

`ListQueues`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListQueues`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest =
        ListQueuesRequest {
            queueNamePrefix = prefix
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
```

```
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
}
```

- Para API obtener más información, consulta ListQueuesla AWS SDKAPIreferencia sobre Kotlin.

`ReceiveMessage`

En el siguiente ejemplo de código, se muestra cómo utilizar `ReceiveMessage`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- Para API obtener más información, consulta ReceiveMessagela AWS SDKAPIreferencia sobre Kotlin.

`SendMessage`

En el siguiente ejemplo de código, se muestra cómo utilizar `SendMessage`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun sendMessages(
    queueUrlVal: String,
    message: String,
) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest =
        SendMessageRequest {
            queueUrl = queueUrlVal
            messageBody = message
            delaySeconds = 10
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessage(sendRequest)
        println("A single message was successfully sent.")
    }
}

suspend fun sendBatchMessages(queueUrlVal: String?) {
    println("Sending multiple messages")

    val msg1 =
        SendMessageBatchRequestEntry {
            id = "id1"
            messageBody = "Hello from msg 1"
        }

    val msg2 =
        SendMessageBatchRequestEntry {
            id = "id2"
```

```
            messageBody = "Hello from msg 2"
        }

    val sendMessageBatchRequest =
        SendMessageBatchRequest {
            queueUrl = queueUrlVal
            entries = listOf(msg1, msg2)
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- Para API obtener más información, consulta SendMessagela AWS SDKAPIreferencia sobre Kotlin.

Escenarios

Crear una aplicación de mensajería

El siguiente ejemplo de código muestra cómo crear una aplicación de mensajería mediante AmazonSQS.

SDKpara Kotlin

Muestra cómo usar Amazon SQS API para desarrollar un Spring REST API que envíe y recupere mensajes.

Para obtener el código fuente completo e instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Comprehend
- Amazon SQS

Publicación de mensajes en colas

En el siguiente ejemplo de código, se muestra cómo:

- Cree un tema (FIFOo noFIFO).
- Suscribir 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

SDKpara Kotlin

> **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 aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
```

```
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following 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.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
    println(
        """
```

```
                In this workflow, you will create an SNS topic and subscribe an SQS
 queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
            """.trimIndent(),
        )
        println(DASHES)

        println(DASHES)
        println(
            """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication and
 message filtering.
                Would you like to work with FIFO topics? (y/n)
            """.trimIndent(),
        )
        useFIFO = input.nextLine()
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true
            println("You have selected FIFO")
            println(
                """ Because you have chosen a FIFO topic, deduplication is supported.
            Deduplication IDs are either set in the message or automatically generated
 from content using a hash function.
            If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
            within the five-minute deduplication interval, is accepted but not
 delivered.
            For more information about deduplication, see https://docs.aws.amazon.com/
sns/latest/dg/fifo-message-dedup.html.""",
            )

            println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
            duplication = input.nextLine()
            if (duplication.compareTo("y") == 0) {
                println("Enter a group id value")
                groupId = input.nextLine()
            } else {
                println("Enter deduplication Id value")
                deduplicationID = input.nextLine()
                println("Enter a group id value")
```

```
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended to
the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
```

```
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/latest/
dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's subscription
 to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
```

```
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following \"tone
\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
 deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)
```

```
    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
 continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
```

```
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
```

```
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
```

```
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
```

```
                "The queue " + queueArnVal + " has been subscribed to the topic " +
 topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
```

```
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }
```

```
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
```

```
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Ejemplos de Step Functions que se utilizan SDK para Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante Kotlin with Step Functions. AWS SDK

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Introducción

Introducción a Step Functions

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Step Functions.

SDKpara Kotlin

> **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 aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)
```

```
    listMachines()
}

suspend fun listMachines() {
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- Para API obtener más información, consulta ListStateMachinesla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Conceptos básicos
- Acciones

Conceptos básicos

Conceptos básicos

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.

SDKpara Kotlin

> **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 aws.sdk.kotlin.services.iam.IamClient
import aws.sdk.kotlin.services.iam.model.CreateRoleRequest
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.CreateActivityRequest
import aws.sdk.kotlin.services.sfn.model.CreateStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DeleteActivityRequest
import aws.sdk.kotlin.services.sfn.model.DeleteStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DescribeExecutionRequest
import aws.sdk.kotlin.services.sfn.model.DescribeStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.GetActivityTaskRequest
import aws.sdk.kotlin.services.sfn.model.ListActivitiesRequest
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest
import aws.sdk.kotlin.services.sfn.model.SendTaskSuccessRequest
import aws.sdk.kotlin.services.sfn.model.StartExecutionRequest
import aws.sdk.kotlin.services.sfn.model.StateMachineType
import aws.sdk.kotlin.services.sfn.paginators.listActivitiesPaginated
import aws.sdk.kotlin.services.sfn.paginators.listStateMachinesPaginated
import com.fasterxml.jackson.databind.JsonNode
import com.fasterxml.jackson.databind.ObjectMapper
import com.fasterxml.jackson.databind.node.ObjectNode
import kotlinx.coroutines.flow.transform
import java.util.Scanner
import java.util.UUID
import kotlin.collections.ArrayList
import kotlin.system.exitProcess

/**
 To run this code example, place the chat_sfn_state_machine.json file into your
 project's resources folder.

 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
```

```
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin code example performs the following tasks:

 1. List activities using a paginator.
 2. List state machines using a paginator.
 3. Creates an activity.
 4. Creates a state machine.
 5. Describes the state machine.
 6. Starts execution of the state machine and interacts with it.
 7. Describes the execution.
 8. Deletes the activity.
 9. Deletes the state machine.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val 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.size != 3) {
        println(usage)
        exitProcess(0)
    }

    val roleName = args[0]
    val activityName = args[1]
    val stateMachineName = args[2]
    val sc = Scanner(System.`in`)
    var action = false
```

```
    val polJSON = """{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "states.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
    }"""

    println(DASHES)
    println("Welcome to the AWS Step Functions example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List activities using a Paginator.")
    listActivitesPagnator()
    println(DASHES)

    println(DASHES)
    println("2. List state machines using a paginator.")
    listStatemachinesPagnator()
    println(DASHES)

    println(DASHES)
    println("3. Create a new activity.")
    val activityArn = createActivity(activityName)
    println("The ARN of the Activity is $activityArn")
    println(DASHES)

    // Get JSON to use for the state machine and place the activityArn value into
it.
    val stream = GetStream()
    val jsonString = stream.getStream()

    // Modify the Resource node.
    val objectMapper = ObjectMapper()
    val root: JsonNode = objectMapper.readTree(jsonString)
    (root.path("States").path("GetInput") as ObjectNode).put("Resource",
activityArn)
```

```
    // Convert the modified Java object back to a JSON string.
    val stateDefinition = objectMapper.writeValueAsString(root)
    println(stateDefinition)

    println(DASHES)
    println("4. Create a state machine.")
    val roleARN = createIAMRole(roleName, polJSON)
    val stateMachineArn = createMachine(roleARN, stateMachineName, stateDefinition)
    println("The ARN of the state machine is $stateMachineArn")
    println(DASHES)

    println(DASHES)
    println("5. Describe the state machine.")
    describeStateMachine(stateMachineArn)
    println("What should ChatSFN call you?")
    val userName = sc.nextLine()
    println("Hello $userName")
    println(DASHES)

    println(DASHES)
    // The JSON to pass to the StartExecution call.
    val executionJson = "{ \"name\" : \"$userName\" }"
    println(executionJson)
    println("6. Start execution of the state machine and interact with it.")
    val runArn = startWorkflow(stateMachineArn, executionJson)
    println("The ARN of the state machine execution is $runArn")
    var myList: List<String>
    while (!action) {
        myList = getActivityTask(activityArn)
        println("ChatSFN: " + myList[1])
        println("$userName please specify a value.")
        val myAction = sc.nextLine()
        if (myAction.compareTo("done") == 0) {
            action = true
        }
        println("You have selected $myAction")
        val taskJson = "{ \"action\" : \"$myAction\" }"
        println(taskJson)
        sendTaskSuccess(myList[0], taskJson)
    }
    println(DASHES)

    println(DASHES)
```

```
    println("7. Describe the execution.")
    describeExe(runArn)
    println(DASHES)

    println(DASHES)
    println("8. Delete the activity.")
    deleteActivity(activityArn)
    println(DASHES)

    println(DASHES)
    println("9. Delete the state machines.")
    deleteMachine(stateMachineArn)
    println(DASHES)

    println(DASHES)
    println("The AWS Step Functions example scenario is complete.")
    println(DASHES)
}

suspend fun listStatemachinesPagnator() {
    val machineRequest =
        ListStateMachinesRequest {
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listStateMachinesPaginated(machineRequest)
            .transform { it.stateMachines?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The state machine ARN is ${obj.stateMachineArn}")
            }
    }
}

suspend fun listActivitesPagnator() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listActivitiesPaginated(activitiesRequest)
```

```
            .transform { it.activities?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The activity ARN is ${obj.activityArn}")
            }
    }
}

suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}

suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}

suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("RUNNING") == 0) {
```

```
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("SUCCEEDED") == 0) {
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
            } else {
                println("The Status is neither running or succeeded")
            }
        }
    }
    println("The Status is $status")
}

suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}

suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}

suspend fun startWorkflow(
    stateMachineArnVal: String?,
```

```
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}

suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}

suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
```

```
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}

suspend fun createIAMRole(
    roleNameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = roleNameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine

- DescribeExecution
- DescribeStateMachine
- GetActivityTask
- ListActivities
- ListStateMachines
- SendTaskSuccess
- StartExecution
- StopExecution

Acciones

`CreateActivity`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateActivity`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Para API obtener más información, consulta CreateActivityla AWS SDKAPIreferencia sobre Kotlin.

`CreateStateMachine`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateStateMachine`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}
```

- Para API obtener más información, consulta CreateStateMachinela AWS SDKAPIreferencia sobre Kotlin.

`DeleteActivity`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteActivity`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}
```

- Para API obtener más información, consulta DeleteActivityla AWS SDKAPIreferencia sobre Kotlin.

`DeleteStateMachine`

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteStateMachine`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
```

```
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- Para API obtener más información, consulta DeleteStateMachinela AWS SDKAPIreferencia sobre Kotlin.

`DescribeExecution`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeExecution`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("RUNNING") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("SUCCEEDED") == 0) {
```

```
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
            } else {
                println("The Status is neither running or succeeded")
            }
        }
    }
    println("The Status is $status")
}
```

- Para API obtener más información, consulta DescribeExecutionla AWS SDKAPIreferencia sobre Kotlin.

`DescribeStateMachine`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeStateMachine`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}
```

- Para API obtener más información, consulta DescribeStateMachinela AWS SDKAPIreferencia sobre Kotlin.

`GetActivityTask`

En el siguiente ejemplo de código, se muestra cómo utilizar `GetActivityTask`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}
```

- Para API obtener más información, consulta GetActivityTaskla AWS SDKAPIreferencia sobre Kotlin.

`ListActivities`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListActivities`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listAllActivites() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listActivities(activitiesRequest)
        response.activities?.forEach { item ->
            println("The activity ARN is ${item.activityArn}")
            println("The activity name is ${item.name}")
        }
    }
}
```

- Para API obtener más información, consulta ListActivitiesla AWS SDKAPIreferencia sobre Kotlin.

`ListExecutions`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListExecutions`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getExeHistory(exeARN: String?) {
```

```
    val historyRequest =
        GetExecutionHistoryRequest {
            executionArn = exeARN
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- Para API obtener más información, consulta ListExecutionsla AWS SDKAPIreferencia sobre Kotlin.

`ListStateMachines`

En el siguiente ejemplo de código, se muestra cómo utilizar `ListStateMachines`.

SDKpara Kotlin

> **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 aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
```

```
suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- Para API obtener más información, consulta ListStateMachinesla AWS SDKAPIreferencia sobre Kotlin.

`SendTaskSuccess`

En el siguiente ejemplo de código, se muestra cómo utilizar `SendTaskSuccess`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
```

```
            taskToken = token
            output = json
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- Para API obtener más información, consulta SendTaskSuccessla AWS SDKAPIreferencia sobre Kotlin.

`StartExecution`

En el siguiente ejemplo de código, se muestra cómo utilizar `StartExecution`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}
```

- Para API obtener más información, consulta StartExecutionla AWS SDKAPIreferencia sobre Kotlin.

Support ejemplos que utilizan SDK para Kotlin

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK for Kotlin with. Support

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Introducción

Hola Support

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Support.

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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 Kotlin example performs the following task:

1. Gets and displays available services.
 */

suspend fun main() {
    displaySomeServices()
}

// Return a List that contains a Service name and Category name.
suspend fun displaySomeServices() {
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is: " + service.name)

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                index++
            }
        }
    }
}
```

- Para API obtener más información, consulta DescribeServicesla AWS SDKAPIreferencia sobre Kotlin.

Temas

- Conceptos básicos
- Acciones

Conceptos básicos

Conceptos básicos

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.

SDKpara Kotlin

> **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 Kotlin code example, set up your development environment,
including your credentials.
```

```
For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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 Kotlin 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.
*/

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <fileAttachment>
    Where:
         fileAttachment - The file can be a simple saved .txt file to use as an
 email attachment.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val fileAttachment = args[0]
    println("***** Welcome to the AWS Support case example scenario.")
    println("***** Step 1. Get and display available services.")
    val sevCatList = displayServices()

    println("***** Step 2. Get and display Support severity levels.")
    val sevLevel = displaySevLevels()
```

```
    println("***** Step 3. Create a support case using the selected service,
 category, and severity level.")
    val caseIdVal = createSupportCase(sevCatList, sevLevel)
    if (caseIdVal != null) {
        println("Support case $caseIdVal was successfully created!")
    } else {
        println("A support case was not successfully created!")
        exitProcess(1)
    }

    println("***** Step 4. Get open support cases.")
    getOpenCase()

    println("***** Step 5. Create an attachment set with a generated file to add to
 the case.")
    val attachmentSetId = addAttachment(fileAttachment)
    println("The Attachment Set id value is $attachmentSetId")

    println("***** Step 6. Add communication with the attachment to the support
 case.")
    addAttachSupportCase(caseIdVal, attachmentSetId)

    println("***** Step 7. List the communications of the support case.")
    val attachId = listCommunications(caseIdVal)
    println("The Attachment id value is $attachId")

    println("***** Step 8. Describe the attachment set included with the
 communication.")
    describeAttachment(attachId)

    println("***** Step 9. Resolve the support case.")
    resolveSupportCase(caseIdVal)

    println("***** Step 10. Get a list of resolved cases for the current day.")
    getResolvedCase()
    println("***** This Scenario has successfully completed")
}

suspend fun getResolvedCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
```

```
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 30
            afterTime = yesterday.toString()
            beforeTime = now.toString()
            includeResolvedCases = true
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}

suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}

suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
```

```
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}

suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
 case")
        } else {
            println("There was an error adding the communication to an AWS Support
 case")
        }
    }
}

suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
```

```
            fileName = myFile.name
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}

suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
```

```
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}

suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}

// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
```

```
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- Para API obtener más información, consulta los siguientes temas en la sección AWS SDKde API referencia sobre Kotlin.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment

- DescribeCases
- DescribeCommunications
- DescribeServices
- DescribeSeverityLevels
- ResolveCase

Acciones

`AddAttachmentsToSet`

En el siguiente ejemplo de código, se muestra cómo utilizar `AddAttachmentsToSet`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

- Para API obtener más información, consulta AddAttachmentsToSetla AWS SDKAPIreferencia sobre Kotlin.

`AddCommunicationToCase`

En el siguiente ejemplo de código, se muestra cómo utilizar `AddCommunicationToCase`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
 case")
        } else {
            println("There was an error adding the communication to an AWS Support
 case")
        }
    }
}
```

- Para API obtener más información, consulta AddCommunicationToCasela AWS SDKAPIreferencia sobre Kotlin.

`CreateCase`

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateCase`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- Para API obtener más información, consulta CreateCasela AWS SDKAPIreferencia sobre Kotlin.

`DescribeAttachment`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeAttachment`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}
```

- Para API obtener más información, consulta DescribeAttachmentla AWS SDKAPIreferencia sobre Kotlin.

`DescribeCases`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeCases`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}
```

- Para API obtener más información, consulta DescribeCasesla AWS SDKAPIreferencia sobre Kotlin.

`DescribeCommunications`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeCommunications`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}
```

- Para API obtener más información, consulta DescribeCommunicationsla AWS SDKAPIreferencia sobre Kotlin.

`DescribeServices`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeServices`.

SDKpara Kotlin

> **Note**
>
> Hay más información. 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.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
```

```
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- Para API obtener más información, consulta DescribeServicesla AWS SDKAPIreferencia sobre Kotlin.

`DescribeSeverityLevels`

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeSeverityLevels`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
```

```
}
```

- Para API obtener más información, consulta DescribeSeverityLevelsla AWS SDKAPIreferencia sobre Kotlin.

`ResolveCase`

En el siguiente ejemplo de código, se muestra cómo utilizar `ResolveCase`.

SDKpara Kotlin

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- Para API obtener más información, consulta ResolveCasela AWS SDKAPIreferencia sobre Kotlin.

Ejemplos de Amazon Translate que utilizan SDK Kotlin

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK uso de Kotlin con Amazon Translate.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

Escenarios

Creación de una SNS aplicación de Amazon

El siguiente ejemplo de código muestra cómo crear una aplicación que tenga funciones de suscripción y publicación y que traduzca los mensajes.

SDKpara Kotlin

Muestra cómo usar Amazon SNS Kotlin API para crear una aplicación que tenga funciones de suscripción y publicación. Además, esta aplicación de ejemplo también traduce los mensajes.

Para obtener el código fuente completo y las instrucciones sobre cómo crear una aplicación web, consulte el ejemplo completo en GitHub.

Para obtener el código fuente completo y las instrucciones sobre cómo crear una aplicación Android nativa, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon SNS
- Amazon Translate

Seguridad para el AWS SDK para Kotlin

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 tú AWS y tú. 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 AWS SDK para Kotlin
- AWS SDK para Kotlin soporte para TLS 1.2
- 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 AWS SDK para Kotlin

El modelo de responsabilidad AWS compartida modelo se aplica a la protección de datos en AWS SDK para Kotlin. Como se describe en este modelo, AWS es responsable de proteger la

infraestructura global que ejecuta todos los Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres 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 la sección Privacidad de datos FAQ. Para obtener información sobre la protección de datos en Europa, consulte el modelo de responsabilidad AWS compartida y la entrada del GDPR blog sobre AWS seguridad.

Para proteger los 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 multifactorial (MFA) con cada cuenta.
- UseSSL/TLSpara comunicarse con AWS los recursos. Necesitamos TLS 1.2 y recomendamos TLS 1.3.
- Configure API y registre la actividad del usuario con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte Cómo trabajar con CloudTrail senderos en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utilice servicios de seguridad gestionados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita entre FIPS 140 y 3 módulos criptográficos validados para acceder a AWS través de una interfaz de línea de comandos o unaAPI, utilice un FIPS terminal. Para obtener más información sobre los FIPS puntos finales disponibles, consulte la Norma federal de procesamiento de información () FIPS 140-3.

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 trabajas con SDK Kotlin o con otros dispositivos Servicios de AWS mediante la consola,API, AWS CLI o. AWS SDKs Cualquier dato que ingrese en etiquetas o campos de texto de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporcionas una URL a un servidor externo, te recomendamos encarecidamente que no incluyas información sobre las credenciales URL para validar tu solicitud a ese servidor.

AWS SDK para Kotlin soporte para TLS 1.2

La siguiente información se aplica únicamente a SSL la implementación de Java (la SSL implementación predeterminada en la AWS SDK para Kotlin versión dirigida aJVM). Si utilizas una SSL implementación diferente, consulta tu SSL implementación específica para aprender a aplicar TLS las versiones.

TLSsoporte en Java

TLSLa versión 1.2 se admite a partir de Java 7.

¿Cómo comprobar la TLS versión

Para comprobar qué TLS versión es compatible con su máquina virtual Java (JVM), puede utilizar el siguiente código.

```
println(SSLContext.getDefault().supportedSSLParameters.protocols.joinToString(separator
 = ", "))
```

Para ver el SSL protocolo de enlace en acción y qué versión de TLS se utiliza, puede utilizar la propiedad del sistema javax.net.debug.

```
-Djavax.net.debug=ssl
```

Identity and Access Management

AWS Identity and Access Management (IAM) es una Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. IAMlos administradores controlan quién puede autenticarse (iniciar sesión) y quién puede autorizarse (tener permisos) para usar AWS los recursos. IAMes una Servicio de AWS 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 AWS de 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 ayudarle a solicitar los permisos correctos al administrador. Si no puede acceder a una función de AWS, consulte Solución de problemas AWS de 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. A continuación, 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 puede usarlo IAM su empresa AWS, consulte la guía del usuario del Servicio de AWS que está utilizando.

IAMadministrador: si es IAM administrador, puede que desee obtener más información sobre cómo puede redactar políticas para administrar el acceso AWS. Para ver ejemplos de políticas AWS basadas en la identidad que puede utilizarIAM, 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 IAM usuario o asumiendo un IAM rol. 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 (IAMIdentity Center), 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 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 incluye 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 la versión 4 de la AWS firma para ver API las solicitudes en la Guía del IAM usuario.

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 multifactorial (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactorial en la Guía del AWS IAM Identity Center usuario y Autenticación AWS multifactorial IAM en la Guía del IAM usuario.

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 utiliza 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 obtener 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 empresarial, 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 utiliza AWS IAM Identity Center. Puede crear usuarios y grupos en IAM Identity Center, o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus aplicaciones Cuentas de AWS . Para obtener información sobre IAM Identity Center, consulte ¿Qué es IAM Identity Center? en la Guía AWS IAM Identity Center del usuario.

Usuarios y grupos de IAM

Un IAMusuario es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos utilizar credenciales temporales en lugar de crear IAM usuarios con credenciales de larga duración, como contraseñas y claves de acceso. Sin embargo, si tiene casos de uso específicos que requieren credenciales a largo plazo con IAM los usuarios, le recomendamos que rote las claves de acceso. Para obtener 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 para grandes conjuntos de usuarios. Por ejemplo, puede asignar un nombre a un grupo IAMAdminsy concederle permisos para administrar IAM los recursos.

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 de larga duración permanentes; no obstante, los roles proporcionan credenciales temporales. Para obtener más información, consulte Casos de uso para IAM usuarios en la Guía del IAM usuario.

Roles de IAM

Un IAMrol es una identidad dentro de ti Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Para asumir temporalmente un IAM rol en el AWS Management Console, puede cambiar de un IAM rol de usuario a uno (consola). Puede asumir un rol llamando a una AWS API operación AWS CLI o o utilizando una operación personalizadaURL. Para obtener más información sobre los métodos de uso de los roles, consulte Métodos para asumir un rol en la Guía del IAM usuario.

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 los permisos para este. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información sobre los roles de la federación, consulte Crear un rol para un proveedor de identidades externo (federación) en la Guía del IAM usuario. Si usa IAM Identity Center, configura un conjunto de permisos. Para controlar a qué pueden acceder sus identidades después de autenticarse, IAM Identity Center correlaciona el conjunto de permisos con un rol en. IAM Para obtener información acerca de los conjuntos de permisos, consulta Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .
- Permisos IAM de usuario temporales: un IAM usuario o rol puede asumir un IAM rol para asumir temporalmente diferentes permisos para una tarea específica.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta obtenga acceso a los recursos de su cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunos Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para conocer la diferencia entre las funciones y las políticas basadas en recursos para el acceso multicuenta, consulta el tema sobre el acceso a los recursos entre cuentas IAM en la Guía del IAM usuario.
- Acceso entre servicios: algunos Servicios de AWS utilizan funciones en otros. Servicios de AWS Por ejemplo, cuando realizas una llamada en un servicio, es habitual 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 IAM usuario o un rol para realizar acciones en AWS ellas, se te considera director. Al utilizar algunos servicios, es posible que realice una acción que, a continuación, inicie otra acción en un servicio diferente. FASutiliza los permisos de la persona principal que llama a an Servicio de AWS, junto con los que solicitan, Servicio de AWS para realizar solicitudes a los servicios descendentes. FASlas solicitudes 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 detallada sobre la política a la hora de realizar FAS solicitudes, consulte Reenviar las sesiones de acceso.
  - Función de servicio: una función de servicio es una IAMfunción que un servicio asume 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 Crear un rol para delegar permisos a un Servicio de AWS en la Guía del IAM usuario.

  - Función vinculada a un 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: puedes usar un IAM rol para administrar las credenciales temporales de las aplicaciones que se ejecutan en una EC2 instancia y que realizan AWS CLI o AWS API solicitan. Es preferible hacerlo de este modo a almacenar claves de acceso dentro de la instancia EC2. Para asignar un AWS rol a una EC2 instancia 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 obtener más información, consulta Usar un IAM rol para conceder permisos a las aplicaciones que se ejecutan en EC2 instancias de Amazon en la Guía del IAM usuario.

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 JSON documentos. Para obtener más información sobre la estructura y el contenido de los documentos de JSON políticas, consulte Descripción general de JSON las políticas en la Guía del IAM usuario.

Los administradores pueden usar AWS JSON políticas 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 agregar las políticas de IAM a los 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 AWS Management Console AWS CLI, el o el AWS API.

Políticas basadas en identidad

Las políticas basadas en la identidad son documentos de política de JSON permisos que se pueden adjuntar a una identidad, como un IAM usuario, 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 información sobre cómo crear una política basada en la identidad, consulte Definir IAM permisos personalizados con políticas administradas por el cliente en la Guía del usuario. IAM

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas gestionadas. 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 funciones de su empresa. Cuenta de AWS Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para saber cómo elegir entre una política gestionada o una política integrada, consulte Elegir entre políticas gestionadas y políticas integradas en la Guía del IAM usuario.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de JSON política que se adjuntan a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de buckets de Amazon S3. En los servicios que admiten políticas basadas en recursos, los gestionadores 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 puede usar políticas AWS administradas desde una política IAM basada en recursos.

Listas de control de acceso () ACLs

Las listas de control de acceso (ACLs) controlan qué responsables (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLsson similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de JSON políticas.

Amazon S3 AWS WAF y Amazon VPC son ejemplos de servicios compatiblesACLs. Para obtener más informaciónACLs, consulte la descripción general de la 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 función avanzada en la que se establecen los permisos máximos que una política basada en la identidad puede conceder a una IAM entidad (IAMusuario o rol). 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 (SCPs): SCPs son JSON políticas que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y administrar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilitas todas las funciones de una organización, puedes aplicar políticas de control de servicios (SCPs) a una o a todas tus cuentas. SCPLimita 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 sobre OrganizationsSCPs, consulte las políticas de control de servicios en la Guía del AWS Organizations usuario.
- Políticas de control de recursos (RCPs): RCPs son JSON políticas que puedes usar para establecer los permisos máximos disponibles para los recursos de tus cuentas sin actualizar las IAM políticas asociadas a cada recurso que poseas. Esto RCP limita los permisos de los recursos en las cuentas de los miembros y puede afectar a los permisos efectivos de las identidades Usuario raíz de la cuenta de AWS, incluidos los permisos, independientemente de si pertenecen a su organización. Para obtener más información sobre Organizations e RCPs incluir una lista de Servicios de AWS ese apoyoRCPs, consulte Políticas de control de recursos (RCPs) en la Guía del AWS Organizations usuario.
- 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 puede proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener 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 se AWS determina 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 IAM usuario.

¿Cómo Servicios de AWS trabajar con IAM

Para obtener una visión general de cómo Servicios de AWS funciona con la mayoría de las IAM funciones, consulte AWS los servicios con los que funcionan IAM en la Guía del IAM usuario.

Para obtener información sobre cómo usar una función específica Servicio de AWS IAM, consulta la sección de seguridad de la Guía del usuario del servicio correspondiente.

Solución de problemas AWS de identidad y acceso

Utilice la siguiente información como ayuda para diagnosticar y solucionar los problemas más comunes que pueden surgir al trabajar con AWS yIAM.

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.

El siguiente ejemplo de error se produce cuando el `mateojackson` IAM usuario intenta usar la consola para ver los detalles de un *`my-example-widget`* recurso ficticio, pero no tiene los `awes:`*`GetWidget`* permisos ficticios.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes: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 `awes:`*`GetWidget`*.

Si necesita ayuda, póngase en contacto con AWS el administrador. El gestionador 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.

Algunas 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 gestionador 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

Puedes crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puedes especificar una persona de confianza para que asuma el rol. En el caso de los servicios que respaldan políticas basadas en recursos o listas de control de acceso (ACLs), puedes usar esas políticas para permitir que las personas accedan a tus recursos.

Para obtener 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 propiedad, consulte Proporcionar acceso a un IAM usuario en otro Cuenta de AWS de su propiedad en la Guía del IAM usuario. Cuentas de AWS
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo permitir el acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del IAM usuario.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticadoPara que su aplicación pueda acceder a s externamente (federación de identidades) en la Guía del usuario de IAM.
- Para saber la diferencia entre usar roles y políticas basadas en recursos para el acceso entre cuentas, consulta el tema Acceso a recursos entre cuentas IAM en la Guía del IAM usuario.

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:

- Cumplimiento de seguridad y gobernanza: en estas guías se explican las consideraciones de arquitectura y se proporcionan pasos para implementar las características de seguridad y cumplimiento.
- Diseñando una arquitectura basada en la HIPAA seguridad y el cumplimiento en Amazon Web Services: en este documento técnico se describe cómo pueden utilizar las empresas AWS para crear HIPAA aplicaciones aptas.

> **Note**
>
> No todos son aptos. Servicios de AWS HIPAA Para obtener más información, consulta la Referencia de servicios HIPAA aptos.

- 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 la perspectiva del cumplimiento. En las guías se resumen las mejores prácticas para garantizar la seguridad Servicios de AWS y se orientan a 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, consulta 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, por ejemplo PCIDSS, cumpliendo 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 consumo 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 API llamadas 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). Necesitamos TLS 1.2 y recomendamos TLS 1.3.
- Cifre suites con perfecto secreto (PFS), como (Ephemeral Diffie-Hellman) o DHE 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 principal de IAM. También puedes 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.

Historial de documentos

En este tema se describen los cambios importantes introducidos en la Guía para AWS SDK para Kotlin desarrolladores a lo largo de su historia.

Cambio	Descripción	Fecha
Actualice los ejemplos de archivos de compilación	Muestra los elementos del archivo de compilación generados en la versión 8.11.1 de Gradle. Muestra el uso de. BOM Inserte enlaces a la última versión de los artefactos.	18 de diciembre de 2024
Añadir tema DynamoDB Mapper (Developer Preview)	Asigne clases a elementos de DynamoDB mediante DynamoDB Mapper (versión preliminar para desarrolladores)	29 de octubre de 2024
Actualizar los nombres de los buckets de Amazon S3	Sumas de verificación de Amazon S3 con AWS SDK para Kotlin	30 de septiembre de 2024
Agregue información al motor OkHttp 4	Especifique un tipo HTTP de motor	26 de septiembre de 2024
Agregar información sobre los puntos de AWS conexión basados en cuentas para DynamoDB	AWS Utilice puntos de conexión basados en cuentas	24 de septiembre de 2024
Agrega un tema de solución de problemas FAQs	Solución de problemas de FAQs	18 de septiembre de 2024
Actualice OpenTelemetry el ejemplo de configuración y la	Observabilidad	2 de mayo de 2024

configuración del proveedor de telemetría global predeterminado		
Proporcione más detalles sobre el proceso de creación del cliente de servicio	Cree un cliente de servicio	14 de marzo de 2024
Añadir tema sobre puntos de acceso multirregionales	Trabaje con los puntos de acceso multirregionales de Amazon S3 mediante el SDK para Kotlin	6 de febrero de 2024
Agregue las instrucciones del catálogo de versiones de Gradle	Catálogo de versiones de Gradle (pestaña)	19 de diciembre de 2023
Versión de disponibilidad general	AWS SDK para Kotlin Guía para desarrolladores	27 de noviembre de 2023
Actualice la sección de configuración de los puntos finales del cliente en función de las actualizaciones SDK	Puntos finales del cliente	25 de agosto de 2023
Sumas de comprobación de Amazon S3	Se ha añadido una sección sobre cómo usar sumas de comprobación flexibles con Amazon S3.	14 de agosto de 2023
Añadir tema de observabilidad	Observabilidad	3 de agosto de 2023
Agrega un tema que aborde los reintentos	Reintentos	7 de julio de 2023
Actualice la sección de configuración del HTTP cliente en función de las SDK actualizaciones	HTTPconfiguración del cliente	6 de junio de 2023

Añadir tema HTTP de prefirma	Prefirmar solicitudes	2 de junio de 2023
Añadir tema de HTTP interceptores	HTTPinterceptores	22 de mayo de 2023
Support para la actualización automática de los tokens	Actualice las instrucciones para el acceso mediante el inicio de sesión único.	18 de mayo de 2023
Sumas de comprobación de Amazon S3	Añada una sección que describa cómo utilizar las sumas de comprobación con Amazon S3.	15 de mayo de 2023
Anule la configuración del cliente del servicio	Agregue una sección que describa cómo anular la configuración de un cliente de servicio y cómo se ven afectados los recursos.	8 de mayo de 2023
Imponga una versión mínima TLS	Agregue una sección que describa las opciones para aplicar una TLS versión mínima.	3 de mayo de 2023
Configuración del punto final del cliente	Agregue un tema que aborde la configuración del punto final del cliente.	7 de abril de 2023
IAMmejores prácticas, actualizaciones	Guía actualizada para adaptarla a las IAM mejores prácticas. Para obtener más información, consulte las mejores prácticas de seguridad en IAM.	22 de marzo de 2023

Agregue un ejemplo de archivo de proyecto de Maven	Muestra un ejemplo de un archivo de proyecto de Maven además de un archivo de proyecto en Gradle en el tema Configuración.	2 de diciembre de 2022
Revise el contenido de la vista previa de la Guía para desarrolladores	El contenido se actualizó para reflejar el trabajo de desarrollo reciente	4 de octubre de 2022
AWS SDK para Kotlin Versión preliminar para desarrolladores	AWS SDK para Kotlin	2 de diciembre de 2021
AWS SDK para Kotlin versión alfa	Anunciamos una nueva versión AWS SDK para Kotlin alfa	30 de agosto de 2021

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.