Developer Guide

AWS SDK for Kotlin

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS SDK for Kotlin: Developer Guide

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is the AWS SDK for Kotlin? 1
Get started with the SDK 1
Maintenance and support for SDK major versions 1
Additional resources 1
Get started 3
Step 1: Set up for this tutorial 3
Step 2: Create the project 3
Step 3: Write the code 6
Step 4: Build and run the application 8
Success 8
Cleanup 9
Next steps 9
Set up 10
Basic set up 10
Overview 10
Sign-in ability to the AWS access portal 11
Configure single sign-on 12
Sign in using the AWS CLI 13
Install Java and a build tool 13
Use temporary credentials 14
Create project build files 15
Code your project 20
Log in using the AWS CLI 20
Configure 22
Create a service client 24
Configure a client in code 25
Configure a client from the environment 25
Close the client 26
AWS Region selection 27
Default Region provider chain 27
Credentials providers 28
Default credentials provider chain 28
Specify a credentials provider 31
Client endpoints 32

Custom configuration ... 32
Examples ... 35
HTTP ... 36
HTTP client configuration ... 37
Use an HTTP proxy ... 40
Interceptors ... 41
Enforce a minimum TLS version ... 42
Retries ... 44
Understanding retry behavior ... 44
Customizing retry behavior ... 48
Observability ... 56
Configure a `TelemetryProvider` ... 56
Metrics ... 58
Logging ... 60
Telemetry providers ... 64
Override client configuration ... 65
Overridden client lifecycle ... 66
Shared resources ... 66
Use the SDK ... 68
Make requests ... 68
Service interface DSL overloads ... 69
Requests with no required inputs ... 70
Coroutines ... 70
Making concurrent requests ... 70
Making blocking requests ... 71
Streaming operations ... 72
Streaming responses ... 72
Streaming requests ... 73
Pagination ... 74
Waiters ... 75
Error handling ... 76
Service exceptions ... 76
Client exceptions ... 76
Error metadata ... 77
Presign requests ... 77
Presigning basics ... 78

Advanced presigning configuration ... 78
Presigning POST and PUT requests ... 79
Operations the SDK can presign ... 80
Troubleshooting FAQs ... 80
How do I fix "connection closed" issues? ... 80
Why are exceptions thrown before reaching the maximum attempts? ... 81
How do I fix `NoSuchMethodError` or NoClassDefFoundError? ... 82
How do I resolve dependency conflicts? ... 83
Mocking ... 84
MockK ... 85
Work with AWS services ... 90
Amazon S3 ... 91
Data integrity protection with checksums ... 92
Work with Multi-Region Access Points ... 96
DynamoDB ... 102
Use AWS account-based endpoints ... 102
Use DynamoDB Mapper (Developer Preview) ... 103
Code examples ... 130
API Gateway ... 131
Scenarios ... 132
Aurora ... 132
Basics ... 133
Actions ... 146
Scenarios ... 132
Auto Scaling ... 160
Basics ... 133
Actions ... 146
Amazon Bedrock ... 177
Actions ... 146
Amazon Bedrock Runtime ... 178
Amazon Nova ... 179
CloudWatch ... 183
Basics ... 133
Actions ... 146
CloudWatch Logs ... 223
Actions ... 146

Amazon Cognito Identity Provider 226
Actions 146
Scenarios 132
Amazon Comprehend 242
Scenarios 132
DynamoDB 242
Basics 133
Actions 146
Scenarios 132
Amazon EC2 271
Basics 133
Actions 146
Amazon ECR 301
Basics 133
Actions 146
OpenSearch Service 330
Actions 146
EventBridge 334
Basics 133
Actions 146
AWS Glue 365
Basics 133
Actions 146
IAM 376
Basics 133
Actions 146
AWS IoT 395
Basics 133
Actions 146
AWS IoT data 419
Actions 146
AWS IoT FleetWise 421
Basics 133
Actions 146
Amazon Keyspaces 456
Basics 133

Actions ... 146
AWS KMS ... 481
Actions ... 146
Lambda ... 491
Basics ... 133
Actions ... 146
Scenarios ... 132
Amazon Location ... 500
Basics ... 133
Actions ... 146
MediaConvert ... 530
Actions ... 146
Amazon Pinpoint ... 535
Actions ... 146
Amazon RDS ... 545
Basics ... 133
Actions ... 146
Scenarios ... 132
Amazon RDS Data Service ... 563
Scenarios ... 132
Amazon Redshift ... 564
Actions ... 146
Scenarios ... 132
Amazon Rekognition ... 568
Actions ... 146
Scenarios ... 132
Route 53 domain registration ... 587
Basics ... 133
Actions ... 146
Amazon S3 ... 606
Basics ... 133
Actions ... 146
Scenarios ... 132
SageMaker AI ... 628
Actions ... 146
Scenarios ... 132

Secrets Manager ... 653
Actions ... 146
Amazon SES ... 654
Scenarios ... 132
Amazon SNS ... 656
Actions ... 146
Scenarios ... 132
Amazon SQS ... 683
Actions ... 146
Scenarios ... 132
Step Functions ... 705
Basics ... 133
Actions ... 146
Support ... 726
Basics ... 133
Actions ... 146
Amazon Translate ... 744
Scenarios ... 132
Security ... 746
Data protection ... 746
Enforcing TLS 1.2 ... 747
TLS support in Java ... 747
How to check the TLS version ... 748
Identity and Access Management ... 748
Audience ... 748
Authenticating with identities ... 749
Managing access using policies ... 752
How AWS services work with IAM ... 755
Troubleshooting AWS identity and access ... 755
Compliance Validation ... 757
Resilience ... 758
Infrastructure Security ... 758
Document history ... 760

What is the AWS SDK for Kotlin?

The AWS SDK for Kotlin provides Kotlin APIs for Amazon Web Services. Using the SDK, you can build Kotlin applications that work with Amazon S3, Amazon EC2, Amazon DynamoDB, and more. With the Kotlin SDK, you can target the JVM platform or Android API level 24 or higher. Support for additional platforms like JavaScript and Native is coming in future releases.

To track upcoming features in the future releases, see our roadmap on GitHub.

Get started with the SDK

To get started with the SDK, follow the Get started tutorial.

To set up your development environment, see Set up.

To create and configure service clients for making requests to AWS services, see Configuration. For information on various features of the SDK, see Use the SDK.

For use cases and examples of performing specific API operations, see Code examples.

Maintenance and support for SDK major versions

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

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

Additional resources

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

- AWS developer blog
- Developer forums
- SDK source (GitHub)
- AWS Code Sample Catalog

- @awsdevelopers (X, formerly Twitter)

Get started with the SDK for Kotlin

The AWS SDK for Kotlin provides Kotlin APIs for each AWS service. Using the SDK, you can build Kotlin applications that work with Amazon S3, Amazon EC2, Amazon DynamoDB, and more.

This tutorial shows you how to use Gradle to define dependencies for the AWS SDK for Kotlin. Then, you create code that writes data to a DynamoDB table. Although you might want to use the features of an IDE, all you need for this tutorial is a terminal window and a text editor.

Follow these steps to complete this tutorial:

- Step 1: Set up for this tutorial
- Step 2: Create the project
- Step 3: Write the code
- Step 4: Build and run the application

Step 1: Set up for this tutorial

Before you begin this tutorial, you need an IAM Identity Center permission set that can access DynamoDB and you need a Kotlin development environment configured with IAM Identity Center single sign-on settings to access to AWS.

Follow the instructions in the Basic set up of this guide to get the basics setup for this tutorial.

After you have configured your development environment with single sign-on access for the Kotlin SDK and you have an active AWS access portal session, continue with Step 2.

Step 2: Create the project

To create the project for this tutorial, first use Gradle to create the basic files for a Kotlin project. Then, update the files with the required settings, dependencies, and code for the AWS SDK for Kotlin.

To create a new project using Gradle

> **Note**
>
> This tutorial uses Gradle version 8.11.1 with the `gradle init` command, which offers five prompts in step 3 below. If you use a different version of Gradle the prompts might differ and as well as the prefilled versions of artifacts.

1. Create a new directory called `getstarted` in a location of your choice, such as your desktop or home folder.
2. Open a terminal or command prompt window and navigate to the `getstarted` directory you created.
3. Use the following command to create a new Gradle project and a basic Kotlin class.

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

   - When prompted for the target `Java version`, press `Enter` (defaults to 21).
   - When prompted with `Project name`, press `Enter` (defaults to the directory name, `getstarted` in this tutorial).
   - When prompted for `application structure`, press `Enter` (defaults to `Single application project`).
   - When prompted with `Select test framework`, press `Enter` (defaults to `kotlin.test`).
   - When prompted with `Generate build using new APIs and behavior`, press `Enter` (defaults to no).

To configure your project with dependencies for the AWS SDK for Kotlin and Amazon S3

- In the `getstarted` directory that you created in the previous procedure, replace the contents of the `settings.gradle.kts` file with the following content, replacing *`X.Y.Z`* with the latest version of the SDK for 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")
```

- Navigate to the `gradle` directory inside of the `getstarted` directory. Replace the contents of the version catalog file named `libs.versions.toml` with the following content:

```
[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" }
```

- Navigate to the app directory and open the `build.gradle.kts` file. Replace its contents with the following code, and then save your changes.

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


dependencies {
    implementation(awssdk.services.s3) // Add dependency on the AWS SDK for 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()
}
```

The `dependencies` section contains an `implementation` entry for the Amazon S3 module of the AWS SDK for Kotlin. The Gradle compiler is configured to use Java 21 in the `java` section.

Step 3: Write the code

After the project has been created and configured, edit the project's default class App to use the following example code.

1. In your project folder app, navigate to the directory `src/main/kotlin/org/example`. Open the App`.kt` file.
2. Replace its contents with the following code and save the file.

```
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!")
```

```
}
```

Step 4: Build and run the application

After the project is created and contains the example class, build and run the application.

1. Open a terminal or command prompt window and navigate to your project directory `getstarted`.
2. Use the following command to build and run your application:

```
gradle run
```

> **Note**
>
> If you get an `IdentityProviderException`, you may not have an active single sign-on session. Run the `aws sso login` AWS CLI command to initiate a new session.

The application calls the createBucket API operation to create a new S3 bucket and then calls putObject to put a new object into the new S3 bucket.

In the `cleanUp()` function at the end, the application deletes the object and then deletes the S3 bucket.

To see the results in the Amazon S3 console

1. In `App.kt`, comment out the line `cleanUp(s3)` in the `runBlocking` section and save the file.
2. Rebuild the project and put a new object into a new S3 bucket by running `gradle run`.
3. Sign in to the Amazon S3 console to view the new object in the new S3 bucket.

After you view the object, delete the S3 bucket.

Success

If your Gradle project built and ran without error, then congratulations. You have successfully built your first Kotlin application using the AWS SDK for Kotlin.

Cleanup

When you are done developing your new application, delete any AWS resources that you created during this tutorial to avoid incurring any charges. You might also want to delete or archive the project folder (`get-started`) that you created in Step 2.

Follow these steps to clean up resources:

- If you commented out the call to the `cleanUp()` function, delete the S3 bucket by using the Amazon S3 console.

Next steps

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

- Additional setup steps to work with the SDK for Kotlin
- Configuration of SDK for Kotlin
- Using the SDK for Kotlin
- Security for the SDK for Kotlin

Set up the AWS SDK for Kotlin

To make requests to AWS services using the AWS SDK for Kotlin, you need the following:

- The ability to sign-in to the AWS access portal
- Permission to use the AWS resources your application needs
- A development environment with the following elements:
  - Shared configuration files that are setup with at least one of the following ways:
    - The `config` file contains IAM Identity Center credentials settings so that the SDK can obtain AWS credentials
    - The `credentials` file contains temporary credentials
  - A build automation tool such as Gradle or Maven
- An active AWS access portal session when you are ready to run your application

In this topic

- Basic set up
- Create project build files
- Code your Kotlin project using the SDK for Kotlin

Basic set up

Overview

To successfully develop applications that access AWS services using the AWS SDK for Kotlin, the following requirements must be met.

- You must be able to sign in to the AWS access portal available in the AWS IAM Identity Center.
- The permissions of the IAM role configured for the SDK must allow access to the AWS services that your application requires. The permissions associated with the **PowerUserAccess** AWS managed policy are sufficient for most development needs.
- A development environment with the following elements:
  - Shared configuration files that are set up in at least one of the following ways:

  - The `config` file contains IAM Identity Center single sign-on settings so that the SDK can get AWS credentials.
  - The `credentials` file contains temporary credentials.
- An installation of Java 8 or later.
- A build automation tool such as Maven or Gradle.
- A text editor to work with code.
- (Optional, but recommended) An IDE (integrated development environment) such as IntelliJ IDEA or Eclipse.

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

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

> **Important**
>
> The instructions in this setup section assume that you or organization uses IAM Identity Center. If your organization uses an external identity provider that works independently of IAM Identity Center, find out how you can get temporary credentials for the SDK for Kotlin to use. Follow these instructions to add temporary credentials to the `~/.aws/credentials` file.
> If your identity provider adds temporary credentials automatically to the `~/.aws/credentials` file, make sure that the profile name is `[default]` so that you do not need to provide a profile name to the SDK or AWS CLI.

Sign-in ability to the AWS access portal

The AWS access portal is the web location where you manually sign in to the IAM Identity Center. The format of the URL is `d-xxxxxxxxxx.awsapps.com/start` or *`your_subdomain`*`.awsapps.com/start`.

If you are not familiar with the AWS access portal, follow the guidance for account access in the IAM Identity Center authentication topic in the AWS SDKs and Tools Reference Guide.

Set up single sign-on access for the SDK

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

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

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

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

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

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

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

Sign in using the AWS CLI

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

```
aws sso login
```

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

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

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

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

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

Install Java and a build tool

Your development environment needs the following:

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

- For information about how to install and use IntelliJ IDEA, see https://www.jetbrains.com/idea/.

Use temporary credentials

As an alternative to configuring IAM Identity Center single sign-on access for the SDK, you can configure your development environment with temporary credentials.

Set up a local credentials file for temporary credentials

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

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

3. Save the file. The file `~/.aws/credentials` should now exist on your local development system. This file contains the [default] profile that the SDK for Kotlin uses if a specific named profile is not specified.
4. Sign in to the AWS access portal
5. Follow these instructions under the Manual credential refresh heading to copy IAM role credentials from the AWS access portal.

   a. For step 4 in the linked instructions, choose the IAM role name that grants access for your development needs. This role typically has a name like **PowerUserAccess** or **Developer**.

   b. For step 7, select the **Manually add a profile to your AWS credentials file** option and copy the contents.

6. Paste the copied credentials into your local `credentials` file and remove the generated profile name. Your file should resemble the following:

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

7. Save the `credentials` file

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

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

Create project build files

After you configure single sign-on access and your development environment, create a Kotlin project using your preferred build tool. In the build file, specify the dependencies for the AWS services that your application needs to access.

To see the list of all possible Maven artifact names, consult the API reference documentation. To find the latest version of the SDK, check the latest release on GitHub.

The following sample build files provide the necessary elements to begin coding a project with seven AWS services.

Gradle

The AWS SDK for Kotlin publishes a Gradle version catalog and bill of materials (BOM) that can help you discover the names of dependencies and keep version numbers synchronized across multiple artifacts.

Note that version catalogs are a preview feature of Gradle before version 8. Depending on the version of Gradle that you use, you may need to opt-in via the Feature Preview API.

To use a Gradle version catalog

1. In your `settings.gradle.kts` file, add a `versionCatalogs` block inside the `dependencyResolutionManagement` block.

   The following example file configures the AWS SDK for Kotlin version catalog. You can navigate to the *`X.Y.Z`* link to see the latest version available.

   ```
   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 dependencies in `build.gradle.kts` by using the type-safe identifiers made available by the version catalog.

   The following example file declares dependencies for seven AWS services.

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

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

repositories {
    mavenCentral()
}

dependencies {
    implementation(platform(awssdk.bom))
    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"
}
```

*Java version, for example 17 or 21.

Maven

The following example `pom.xml` file has dependencies for seven AWS services. You can navigate to the *X.Y.Z* link to see the latest version available.

```
<?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>${kotlin.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>${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>
```

*Java version, for example 17 or 21.

Code your Kotlin project using the SDK for Kotlin

Now the fun begins. As you develop your application, you can refer to the AWS SDK for Kotlin API Reference for complete information on the API operations. Use the following links for general Kotlin API information:

- Standard library API reference
- Coroutines overview
- Coroutines API

Log in using the AWS CLI

Whenever you run a program that accesses AWS services, you need an active AWS access portal session. You do this by running the following command.

```
aws sso login
```

Since you have a default profile setup, you do not need to call the command with a `--profile` option. If your IAM Identity Center single sign-on configuration uses a named profile, the command is `aws sso login --profile named-profile`.

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

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

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

Configure the AWS SDK for Kotlin

This section explains how to configure a service client using the AWS SDK for Kotlin. For more information, see the SDK and Tools Reference Guide, which includes an overview of configuration that applies to all AWS SDKs.

Contents

- Create a service client
  - Configure a client in code
  - Configure a client from the environment
  - Close the client
- AWS Region selection
  - Default Region provider chain
- Credentials providers
  - The default credentials provider chain
    - Learn about the default credentials provider chain
  - Specify a credentials provider
    - Cache credentials with a standalone provider
- Configure client endpoints
  - Custom configuration
    - Set endpointUrl
    - Set endpointProvider
      - EndpointProvider properties
    - endpointUrl or endpointProvider
    - A note about Amazon S3
  - Examples
    - endpointUrl example
    - endpointProvider example
    - endpointUrl and endpointProvider
- HTTP
  - HTTP client configuration

- Basic configuration
  - Imports
  - Code
- Specify an HTTP engine type
  - Imports
  - Code
  - Use the OkHttp4Engine
  - Use an explicit HTTP client
    - Imports
    - Code
- Use an HTTP proxy
  - Use JVM system properties
  - Use environment variables
  - Use a proxy on EC2 instances
- HTTP interceptors
  - Interceptor registration
    - Interceptor for all service client operations
    - Interceptor for only specific operations
- Enforce a minimum TLS version
  - Configure the HTTP engine
  - Set the sdk.minTls JVM system property
  - Set the SDK_MIN_TLS environment variable
- Retries in the AWS SDK for Kotlin
- Understanding retry behavior
  - Default retry configuration
  - Which exceptions are retryable?
    - Retryable by error code
    - Retryable by HTTP status code
    - Retryable by error type
    - Retryable by SDK metadata

        - Check if an exception is retryable
        - What exceptions reach your code when retries fail
    - Customizing retry behavior
        - Configure maximum attempts
        - Configure delays and backoff
        - Configure retry token bucket
        - Configure adaptive retries
- Observability
    - Configure a TelemetryProvider
        - Configure the default global telemetry provider
        - Configure a telemetry provider for a specific service client
    - Metrics
    - Logging
        - Specify log mode for wire-level messages
            - Set log mode in code
            - Set log mode from the environment
    - Telemetry providers
        - Configure the OpenTelemetry-based telemetry provider
            - Prerequisites
            - Configure the SDK
            - Resources
- Override service client configuration
    - Lifecycle of an overridden client
    - Resources shared between clients

Create a service client

To make a request to an AWS service, you must first instantiate a client for that service.

You can configure common settings for service clients, such as the HTTP client to use, logging level, and retry configuration. Additionally, each service client requires an AWS Region and a

credentials provider. The SDK uses these values to send requests to the correct Region and to sign requests with the correct credentials.

You can specify these values programmatically in code or have them automatically loaded from the environment.

Configure a client in code

To configure a service client with specific values, you can specify them in a lambda function passed to the service client factory method as shown in the following snippet.

```
val dynamoDbClient = DynamoDbClient {
    region = "us-east-1"
    credentialsProvider = ProfileCredentialsProvider(profileName = "myprofile")
}
```

Any values that you don't specify in the configuration block are set to defaults. For instance, if you don't specify a credentials provider as the previous code does, the credentials provider defaults to the default credentials provider chain.

> **Warning**
>
> Some properties such as `region` do not have a default. You must specify them explicitly in the configuration block when you use programmatic configuration. If the SDK can't resolve the property, API requests may fail.

Configure a client from the environment

When creating a service client, the SDK can inspect locations in the current execution environment to determine some configuration properties. Those locations include shared config and credentials files, environment variables, and JVM system properties. The properties available to be resolved include AWS Region, retry strategy, log mode, and others. For more information on all the settings that the SDK can resolve from the execution environment, see the AWS SDKs and Tools Settings Reference Guide.

To create a client with environment-sourced configuration, use the static method `suspend fun fromEnvironment()` on the service client interface:

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

Creating a client this way is useful when running on Amazon EC2, AWS Lambda, or any other context where the configuration of a service client is available from the environment. This decouples your code from the environment that it's running in and makes it easier to deploy your application to multiple Regions without changing the code.

Additionally, you can override specific properties by passing a lambda block to `fromEnvironment`. The following example loads some configuration properties from the environment (e.g., Region) but specifically overrides the credentials provider to use credentials from a profile.

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

The SDK uses default values for any configuration property that can't be determined from programmatic settings or from the environment. For instance, if you don't specify a credentials provider in code or in an environment setting, the credentials provider defaults to the default credentials provider chain.

Warning

Some properties such as Region do not have a default. You must specify them in an environment setting or explicitly in the configuration block. If the SDK can't resolve the property, API requests may fail.

Note

Although credentials-related properties—such as temporary access keys and SSO configuration—can be found in the execution environment, the values are not sourced by the client at creation time. Instead, the values are accessed by the credentials provider layer at each request.

Close the client

When you no longer need the service client, close it to release any resources that it's using:

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

Because service clients extend the `Closeable` interface, you can use the `use` extension to close the client automatically at the end of a block as shown in the following snippet.

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

In the previous example, the lambda block receives a reference to the client that was just created. You can invoke operations on this client reference and when the block is completed— including by throwing an exception—the client is closed.

AWS Region selection

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

Default Region provider chain

When loading a service client's configuration from the environment, the following lookup process is used:

1. Any explicit Region set on the builder.
2. The `aws.region` JVM system property is checked. If it's set, that Region is used in the configuration of the client.
3. The `AWS_REGION` environment variable is checked. If it's set, that Region is used in the configuration of the client.
   a. **Note**: This environment variable is set by the Lambda container.
4. The SDK checks the AWS shared configuration file. If the `region` property is set for the active profile, the SDK uses it.
   a. The `AWS_CONFIG_FILE` environment variable can be used to customize the location of the shared config file.

   b. The `aws.profile` JVM system property or the `AWS_PROFILE` environment variable can be used to customize the profile that the SDK loads.
5. The SDK attempts to use the Amazon EC2 Instance Metadata Service to determine the Region of the currently running EC2 instance.
6. If the Region still isn't resolved at this point, client creation fails with an exception.

Credentials providers

> ⚠ **The order in which the default credential provider chain resolves credentials changed with version 1.4.0 For details, see the note below.**

When you send requests to Amazon Web Services using the AWS SDK for Kotlin, requests must be cryptographically signed with credentials issued by AWS. The Kotlin SDK signs the request automatically for you. To acquire the credentials, the SDK can use configuration settings that are located in several places, for example JVM system properties, environment variables, shared AWS `config` and `credentials` files, and Amazon EC2 instance metadata.

The SDK uses the *credentials provider* abstraction to simplify the process of retrieving credentials from various sources. The SDK contains several credentials provider implementations.

For example, If the retrieved configuration includes IAM Identity Center single sign-on access settings from the shared `config` file, the SDK works with the IAM Identity Center to retrieve temporary credentials that it uses to make request to AWS services. With this approach to acquiring credentials, the SDK uses the IAM Identity Center provider (also known as the SSO credentials provider). The set up section of this guide described this configuration.

To use a specific credentials provider, you can specify one when you create a service client. Alternatively, you can use the default credentials provider chain to search for configuration settings automatically.

The default credentials provider chain

When not explicitly specified at client construction, the SDK for Kotlin uses a credentials provider that sequentially checks each place where you can supply credentials. This default credentials provider is implemented as a chain of credentials providers.

To use the default chain to supply credentials in your application, create a service client without explicitly providing a `credentialsProvider` property.

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

For more information about service client creation, see construct and configure a client.

Learn about the default credentials provider chain

The default credentials provider chain searches for credentials configuration using the following predefined sequence. When the configured settings provide valid credentials, the chain stops.

1. **AWS access keys (JVM system properties)**

   The SDK looks for the `aws.accessKeyId`, `aws.secretAccessKey`, and `aws.sessionToken` JVM system properties.

2. **AWS access keys (environment variables)**

   The SDK attempts to load credentials from the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`, and `AWS_SESSION_TOKEN` environment variables.

3. **Web identity token**

   The SDK looks for the environment variables `AWS_WEB_IDENTITY_TOKEN_FILE` and `AWS_ROLE_ARN` (or the JVM system properties `aws.webIdentityTokenFile` and `aws.roleArn`). Based on the token information and the role, the SDK acquires temporary credentials.

4. **A profile in a configuration file**

   In this step, the SDK uses settings associated with a profile. By default, the SDK uses the shared AWS `config` and `credentials` files, but if the `AWS_CONFIG_FILE` environment variable is set, the SDK uses that value. If the `AWS_PROFILE` environment variable (or `aws.profile` JVM system property) is *not* set, the SDK looks for the “default” profile, otherwise it looks for the profile that matches `AWS_PROFILE`’s value.

   The SDK looks for the profile based on the configuration described in the previous paragraph and uses the settings defined there. If the settings found by the SDK contain a mix of settings for different credential-provider approaches, the SDK uses the following ordering:

a. AWS access keys (configuration file) - The SDK uses the settings for `aws_access_key_id`, `aws_access_key_id`, and `aws_session_token`.

b. Assume role configuration - If the SDK finds `role_arn` and `source_profile` or `credential_source` settings, it attempts to assume a role. If the SDK finds the `source_profile` setting, it sources credentials from another profile to receive temporary credentials for the role specified by `role_arn`. If the SDK finds the `credential_source` setting, it sources credentials from an Amazon ECS container, an Amazon EC2 instance, or from environment variables depending on the value of the `credential_source` setting. It then uses those credentials to acquire temporary credentials for the role.

   A profile should contain either the `source_profile` setting or the `credential_source` setting, but not both.

c. Web identity token configuration - If the SDK finds `role_arn` and `web_identity_token_file` settings, it acquires temporary credentials to access AWS resources based on the `role_arn` and the token.

d. SSO token configuration - If the SDK finds `sso_session`, `sso_account_id`, `sso_role_name` settings (along with a companion `sso-session` section in the configuration files), the SDK retrieves temporary credentials from the IAM Identity Center service.

e. Legacy SSO configuration - If the SDK finds `sso_start_url`, `sso_region`, `sso_account_id`, and `sso_role_name` settings, the SDK retrieves temporary credentials from the IAM Identity Center service.

f. Process configuration - If the SDK finds a `credential_process` setting, it uses the path value to invoke a process and acquire temporary credentials.

5. Container credentials

The SDK looks for environment variables `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` or `AWS_CONTAINER_CREDENTIALS_FULL_URI` and `AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE` or `AWS_CONTAINER_AUTHORIZATION_TOKEN`. It uses these values to load credentials from the specified HTTP endpoint through a GET request.

6. IMDS credentials

The SDK attempts to fetch credentials from the Instance Metadata Service at the default or configured HTTP endpoint. The SDK only supports IMDSv2.

If credentials still aren't resolved at this point, client creation fails with an exception.

> **Note: Change in the order of credentials resolution**
>
> The ordering of credentials resolution described above is current for the `1.4.x+` release of the SDK for Kotlin. Before the `1.4.0` release, items number 3 and 4 were switched and the current 4a item followed the current 4f item.

Specify a credentials provider

You can specify a credentials provider instead of using the default provider chain. This approach gives you direct control over which credentials the SDK uses.

For example, to use the credentials for an assumed IAM role, specify an `StsAssumeRoleCredentialsProvider` when you create the client:

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

You can also create a custom chain (`CredentialsProviderChain`) that combines multiple providers in your preferred order.

Cache credentials with a standalone provider

> **Important**
>
> The default chain caches credentials automatically. Standalone providers don't cache credentials. To avoid fetching credentials on every API call, wrap your provider with a `CachedCredentialsProvider`. The cached provider fetches new credentials only when current ones expire.

To cache credentials with a standalone provider, use the `CachedCredentialsProvider` class:

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

```
    credentialsProvider =
CachedCredentialsProvider(StsAssumeRoleCredentialsProvider())
}
```

Alternatively, use the `cached()` extension function for more concise code:

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

Configure client endpoints

When the AWS SDK for Kotlin calls an AWS service, one of its first steps is to determine where to route the request. This process is known as endpoint resolution.

You can configure endpoint resolution for the SDK when you build a service client. The default configuration for endpoint resolution is usually fine, but there are several reasons that might lead you to modify the default configuration. Two example reasons are as follows:

- Make requests to a prerelease version of a service or to a local deployment of a service.
- Access to specific service features not yet modeled in the SDK.

> **Warning**
>
> Endpoint resolution is an advanced SDK topic. If you change the default settings, you risk breaking your code. The default settings should apply to most users in production environments.

Custom configuration

You can customize endpoint resolution of a service client with two properties that are available when you build the client:

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

Set `endpointUrl`

You can set a value for `endpointUrl` to indicate a "base" hostname for the service. This value, however, is not final since it is passed as a parameter to the client's `EndpointProvider` instance. The `EndpointProvider` implementation then can inspect and potentially modify that value to determine the final endpoint.

As an example, if you specify an `endpointUrl` value for an Amazon Simple Storage Service (Amazon S3) client and perform a `GetObject` operation, the default endpoint provider implementation injects the bucket name into the hostname value.

In practice, users set an `endpointUrl` value to point at a development or preview instance of a service.

Set `endpointProvider`

A service client's `EndpointProvider` implementation determines final endpoint resolution. The `EndpointProvider` interface shown in the following code block exposes the `resolveEndpoint` method.

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

A service client calls the `resolveEndpoint` method for every request. The service client uses the `Endpoint` value returned by the provider with no further changes.

`EndpointProvider` properties

The `resolveEndpoint` method accepts a service-specific `EndpointParameters` object that contains properties used in endpoint resolution.

Every service includes the following base properties.

Name	Type	Description
`region`	String	The client's AWS Region
`endpoint`	String	A string representation of the value set of `endpointUrl`

Name	Type	Description
`useFips`	Boolean	Whether FIPS endpoints are enabled in the client's configuration
`useDualStack`	Boolean	Whether dual-stack endpoints are enabled in the client's configuration

Services can specify additional properties required for resolution. For example, Amazon S3 `S3EndpointParameters` includes the bucket name and also several Amazon S3-specific feature settings. For example, the `forcePathStyle` property determines whether virtual host addressing can be used.

If you implement your own provider, you shouldn't need to construct your own instance of `EndpointParameters`. The SDK provides the properties for each request and passes them to your implementation of `resolveEndpoint`.

`endpointUrl` or `endpointProvider`

It is important to understand that the following two statements do NOT produce clients with equivalent endpoint resolution behavior:

```
// 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")
    }
}
```

The statement that sets the `endpointUrl` property specifies a *base* URL that is passed to the (default) provider, which can be modified as part of endpoint resolution.

The statement that sets the `endpointProvider` specifies the *final* URL the `S3Client` uses.

Although you can set both properties, in most cases that need customization, you provide one of them. As a general SDK user, you most often provide an `endpointUrl` value.

A note about Amazon S3

Amazon S3 is a complex service with many of its features modeled through customized endpoints customizations, such as bucket virtual hosting. Virtual hosting is a feature of Amazon S3 where the bucket name is inserted into the hostname.

Because of this, we recommend that you don't replace the `EndpointProvider` implementation in an Amazon S3 service client. If you need to extend its resolution behavior, perhaps by sending requests to a local development stack with additional endpoint considerations, we recommend wrapping the default implementation. The following `endpointProvider` example shows a sample implementation of this approach.

Examples

`endpointUrl` example

The following code snippet shows how the general service endpoint can be overridden for an Amazon S3 client.

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

`endpointProvider` example

The following code snippet shows how to provide a custom endpoint provider that wraps the default implementation for 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` and `endpointProvider`

The following example program demonstrates the interaction between the `endpointUrl` and `endpointProvider` settings. This is an advanced use case.

```
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

This section covers the configuration of HTTP-related settings in the AWS SDK for Kotlin.

Topics

- HTTP client configuration
- Use an HTTP proxy
- HTTP interceptors
- Enforce a minimum TLS version

HTTP client configuration

By default, the AWS SDK for Kotlin uses an HTTP client based on OkHttp. You can override the HTTP client and its configuration by supplying an explicitly configured client.

> **Warning**
>
> Regardless of which HTTP engine you use, other dependencies in your project might have transitive dependencies that conflict with the specific engine version required by the SDK. In particular, frameworks such as Spring Boot are known to manage dependencies like OkHttp and to rely on older versions than the SDK. Please see the section called "How do I resolve dependency conflicts?" for more information.

> **Note**
>
> By default, each service client uses its own copy of an HTTP client. If you use multiple services in your application, you might want to construct a single HTTP client and share it across all service clients.

Basic configuration

When you configure a service client, you can configure the default engine type. The SDK manages the resulting HTTP client engine and automatically closes it when it is no longer needed.

The following example shows configuration of an HTTP client during the initialization of a DynamoDB client.

Imports

```
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
```

```
import kotlin.time.Duration.Companion.seconds
```

Code

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

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

Specify an HTTP engine type

For more advanced use cases, you can pass an additional parameter to `httpClient` that specifies the engine type. This way, you can set configuration parameters that are unique to that engine type.

The following example specifies the `OkHttpEngine` that you can use to configure the `maxConcurrencyPerHost` property.

Imports

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

Code

```
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.
}
```

The possible values for the engine type are `OkHttpEngine`, `OkHttp4Engine`, and `CrtHttpEngine`.

To use configuration parameters specific to an HTTP engine, you must add the engine as a compile-time dependency. For the `OkHttpEngine`, you add the following dependency using Gradle.

(You can navigate to the *X.Y.Z* link to see the latest version available.)

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

For the `CrtHttpEngine`, add the following dependency.

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

Use the `OkHttp4Engine`

Use the `OkHttp4Engine` if you can't use the default `OkHttpEngine`. The smithy-kotlin GitHub repository has information about how you configure and use the `OkHttp4Engine`.

Use an explicit HTTP client

When you use an explicit HTTP client, you're responsible for its lifetime, including closing when you no longer need it. An HTTP client must live at least as long as any service client that uses it.

The following code example shows code that keeps the HTTP client stays alive while the `DynamoDbClient` is active. The `use` function makes sure the HTTP client closes properly.

Imports

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

```
import kotlin.time.Duration.Companion.seconds
```

Code

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

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

Use an HTTP proxy

To access AWS through proxy servers using the AWS SDK for Kotlin, you can configure either JVM system properties or environment variables. If both are provided, the JVM system properties take precedence.

Use JVM system properties

The SDK looks for the JVM system properties `https.proxyHost`, `https.proxyPort`, and `http.nonProxyHosts`. For more information on these common JVM system properties, see Networking and Proxies in the Java documentation.

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

Use environment variables

The SDK looks for the `https_proxy`, `http_proxy`, and `no_proxy` environment variables (and capitalized versions of each).

```
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
```

Use a proxy on EC2 instances

If you configure a proxy on an EC2 instance launched with an attached IAM role, make sure to exempt the address that's used to access the instance metadata. To do this, set the `http.nonProxyHosts` JVM system property or `no_proxy` environment variable to the IP address of the Instance Metadata Service, which is `169.254.169.254`. This address does not vary.

```
export no_proxy=169.254.169.254
```

HTTP interceptors

You can use interceptors to hook into the execution of API requests and responses. Interceptors are open-ended mechanisms in which the SDK calls code that you write to inject behavior into the request/response lifecycle. This way, you can modify an in-flight request, debug request processing, view exceptions, and more.

The following example shows a simple interceptor that adds an additional header to all outgoing requests before the retry loop is entered.

```
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()
    }
}
```

For more information and the available interception hooks, see the Interceptor interface.

Interceptor registration

You register interceptors when you construct a service client or when you override configuration for a specific set of operations.

Interceptor for all service client operations

The following code adds an `AddHeader` instance to the interceptors property of the builder. This addition adds the `x-foo-version` header to all operations before the retry loop is entered.

```
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 for only specific operations

By using the `withConfig` extension, you can override service client configuration for one or more operations for any service client. With this capability, you can register additional interceptors for a subset of operations.

The following example overrides the configuration of the `s3` instance for operations within the `use` extension. Operations called on `s3Scoped` contain both the `x-foo-version` and the `x-bar-version` headers.

```
// '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.
}
```

Enforce a minimum TLS version

With the AWS SDK for Kotlin, you can configure the minimum TLS version when you connect to service endpoints. The SDK offers different configuration options. In order of highest to lowest precedence, the options are:

- Explicitly configure the HTTP engine
- Set the `sdk.minTls` JVM system property
- Set the `SDK_MIN_TLS` environment variable

Configure the HTTP engine

When you specify a non-default HTTP engine for a service client, you can set the `tlsContext.minVersion` field.

The following example configures the HTTP engine and any service client that uses it to use TLS v1.2 at a minimum.

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

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

Set the `sdk.minTls` JVM system property

You can set the `sdk.minTls` JVM system property. When you launch an application with the system property set, all HTTP engines constructed by the AWS SDK for Kotlin use the specified minimum TLS version by default. However, you can explicitly override this in the HTTP engine configuration. The allowable values are:

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

Set the `SDK_MIN_TLS` environment variable

You can set the `SDK_MIN_TLS` environment variable. When you launch an application with the environment variable set, all HTTP engines constructed by the AWS SDK for Kotlin use the specified minimum TLS version, unless overridden by another option.

The allowable values are:

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

Retries in the AWS SDK for Kotlin

Calls to AWS services occasionally return unexpected exceptions. Certain types of errors, such as throttling or transient errors, might be successful if the call is retried.

This page describes how the AWS SDK for Kotlin handles retries automatically and how to customize retry behavior for your applications.

Understanding retry behavior

The following sections explain how the SDK determines when to retry requests and what exceptions are considered retryable.

Default retry configuration

By default, every service client is automatically configured with a standard retry strategy. The default configuration tries a call that fails up to three times (the initial attempt plus two retries). The intervening delay between each call is configured with exponential backoff and random jitter to avoid retry storms. This configuration works for the majority of use cases but may be unsuitable in some circumstances, such as high-throughput systems.

The SDK attempts retries only on retryable errors. Examples of retryable errors are socket timeouts, service-side throttling, concurrency or optimistic lock failures, and transient service errors. Missing or invalid parameters, authentication/security errors, and misconfiguration exceptions are not considered retryable.

You can customize the standard retry strategy by setting the maximum attempts, delays and backoff, and token bucket configuration.

Which exceptions are retryable?

The AWS SDK for Kotlin uses a preconfigured retry policy that determines which exceptions are retryable. Service client configuration has a `retryPolicy` property that specifies the policy applied to retries. If no custom value is specified, the default value is AwsRetryPolicy.

The following exceptions are determined to be retryable by `AwsRetryPolicy`:

Retryable by error code

Any `ServiceException` with an `sdkErrorMetadata.errorCode` of:

- `BandwidthLimitExceeded`
- `EC2ThrottledException`
- `IDPCommunicationError`
- `LimitExceededException`
- `PriorRequestNotComplete`
- `ProvisionedThroughputExceededException`
- `RequestLimitExceeded`
- `RequestThrottled`
- `RequestThrottledException`
- `RequestTimeout`
- `RequestTimeoutException`
- `SlowDown`
- `ThrottledException`
- `Throttling`
- `ThrottlingException`
- `TooManyRequestsException`
- `TransactionInProgressException`

Retryable by HTTP status code

Any `ServiceException` with an `sdkErrorMetadata.statusCode` of:

- 500 (Internal Service Error)
- 502 (Bad Gateway)
- 503 (Service Unavailable)
- 504 (Gateway Timeout)

Retryable by error type

Any `ServiceException` with an `sdkErrorMetadata.errorType` of:

- `ErrorType.Server` (such as internal service errors)
- `ErrorType.Client` (such as an invalid request, a resource not found, access denied, etc.)

Retryable by SDK metadata

Any `SdkBaseException` where:

- `sdkErrorMetadata.isRetryable` is `true` (such as a client-side timeout, networking/socket error, etc.)
- `sdkErrorMetadata.isThrottling` is `true` (such as making too many requests in a short amount of time)

For a complete list of exceptions that may be thrown by each service client, consult service-specific API reference documentation.

Check if an exception is retryable

To determine if the SDK considers an exception retryable, check the `isRetryable` property on caught exceptions:

```
try {
    dynamoDbClient.putItem {
        tableName = "MyTable"
        item = mapOf("id" to AttributeValue.S("123"))
    }
} catch (e: SdkBaseException) {
    println("Exception occurred: ${e.message}")

    if (e.sdkErrorMetadata.isRetryable) {
        println("This exception is retryable - SDK will automatically retry")
        println("If you're seeing this, retries may have been exhausted")
    } else {
        println("This exception is not retryable - fix the underlying issue")

        // Common non-retryable scenarios.
        when {
            e.message?.contains("ValidationException") == true ->
```

```
                println("Check your request parameters")
            e.message?.contains("AccessDenied") == true ->
                println("Check your IAM permissions")
            e.message?.contains("ResourceNotFound") == true ->
                println("Verify the resource exists")
        }
    }
}
```

What exceptions reach your code when retries fail

When the SDK's retry mechanism cannot resolve an issue, exceptions are thrown to your application code. Understanding these exception types helps you implement appropriate error handling. These are *not* the exceptions that trigger retries—those are handled internally by the SDK.

Your code will catch the following types of exceptions when retries are exhausted or disabled:

Service exceptions after retry exhaustion

When all retry attempts fail, your code catches the final service exception (subclass of `AwsServiceException`) that caused the last retry attempt to fail. This could be a throttling error, server error, or other service-specific exception that the SDK could not resolve through retries.

Network exceptions after retry exhaustion

When network issues persist through all retry attempts, your code catches `ClientException` instances for problems like connection timeouts, DNS resolution failures, and other connectivity issues that the SDK could not resolve.

Use the following pattern to handle these exceptions in your application:

```
try {
    s3Client.getObject {
        bucket = "amzn-s3-demo-bucket"
        key = "my-key"
    }
} catch (e: AwsServiceException) {
    // Service-side errors that persisted through all retries.
    println("Service error after retries: ${e.errorDetails?.errorCode} - ${e.message}")
```

```
    // Handle specific service errors that couldn't be resolved.
    if (e.errorDetails?.errorCode == "ServiceQuotaExceededException" ||
        e.errorDetails?.errorCode == "ThrottlingException") {
        println("Rate limiting persisted - consider longer delays or quota increase")
    }
} catch (e: ClientException) {
    // Client-side errors (persistent network issues, DNS resolution failures, etc.)
    println("Client error after retries: ${e.message}")
}
```

Customizing retry behavior

The following sections show how to customize the SDK's retry behavior for your specific use case.

Configure maximum attempts

You can customize the default maximum attempts (3) in the `retryStrategy` DSL block during client construction.

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

With the DynamoDB service client shown in the previous snippet, the SDK tries API calls that fail up to five times (the initial attempt plus four retries).

You can disable automatic retries completely by setting the maximum attempts to one as shown in the following snippet.

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

Configure delays and backoff

If a retry is necessary, the default retry strategy waits before it makes the subsequent attempt. The delay for the first retry is small but it grows exponentially for later retries. The maximum amount of delay is capped so that it does not grow too large.

Finally, random jitter is applied to the delays between all attempts. The jitter helps mitigate the effect of large fleets that can cause retry storms. (See this AWS Architecture Blog post for a deeper discussion about exponential backoff and jitter.)

Delay parameters are configurable in the `delayProvider` DSL block.

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

With the configuration shown in the previous snippet, the client delays the first retry attempt for up to 100 milliseconds. The maximum amount of time between any retry attempt is 5 seconds.

The following parameters are available for tuning delays and backoff.

Parameter	Default value	Description
`initialDelay`	10 milliseconds	The maximum amount of delay for the first retry. When jitter is applied, the actual amount of delay may be less.
`jitter`	1.0 (full jitter)	The maximum amplitude by which to randomly reduce the calculated delay. The default value of 1.0 means that the calculated delay can be reduced to any amount up to 100% (for example, down to 0). A value of 0.5 means that the calculated delay can be reduced by up to half. Thus, a max delay of 10ms could be reduced to anywhere

Parameter	Default value	Description
		between 5ms and 10ms. A value of 0.0 means that no jitter is applied. **Important** #Jitter configura tion is an advanced feature. Customizing this behavior is not normally recommend ed.
`maxBackoff`	20 seconds	The maximum amount of delay to apply to any attempt. Setting this value limits the exponential growth that occurs between subsequent attempts and prevents the calculated maximum from being too large. This parameter limits the calculated delay *before* jitter is applied. If applied, jitter might reduce the delay even further.

Parameter	Default value	Description
`scaleFactor`	1.5	The exponential base by which subsequent maximum delays will be increased . For example, given an `initialDelay` of 10ms and a `scaleFactor` of 1.5, the following max delays would be calculated: • Retry 1: 10ms × 1.5^0 = 10ms • Retry 2: 10ms × 1.5^1 = 15ms • Retry 3: 10ms × 1.5^2 = 22.5ms • Retry 4: 10ms × 1.5^3 = 33.75ms When jitter is applied, the actual amount of each delay might be less.

Configure retry token bucket

You can further modify the behavior of the standard retry strategy by adjusting the default token bucket configuration. The retry token bucket helps to reduce retries that are less likely to succeed or that might take more time to resolve, such as timeout and throttling failures.

> **Important**
>
> Token bucket configuration is an advanced feature. Customizing this behavior is not normally recommended.

Each retry attempt (optionally including the initial attempt) decrements some capacity from the token bucket. The amount decremented depends on the type of attempt. For example, retrying transient errors might be cheap, but retrying timeout or throttling errors might be more expensive.

A successful attempt returns capacity to the bucket. The bucket may not be incremented beyond its maximum capacity nor decremented below zero.

Depending on the value of the `useCircuitBreakerMode` setting, attempts to decrement capacity below zero result in one of the following outcomes:

- If the setting is TRUE, an exception is thrown– For example, if too many retries have occurred and more retries are unlikely to succeed.
- If the setting is FALSE, there is a delay – For example, delays until the bucket has sufficient capacity again.

> **Note**
>
> When the circuit breaker activates (token bucket reaches zero capacity), the SDK throws a `ClientException` with the message "Retry capacity exceeded". This is a client-side exception, not an `AwsServiceException`, because it originates from the SDK's retry logic rather than the AWS service. The exception is thrown immediately without attempting the operation, helping prevent retry storms during service outages.

The token bucket parameters are configurable in the `tokenBucket` DSL block:

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

The following parameters are available for tuning the retry token bucket:

Parameter	Default value	Description
`initialTryCost`	0	The amount to decrement from the bucket for initial attempts. The default value of 0 means that no capacity will be decremented and thus initial attempts are not stopped or delayed.
`initialTrySuccessIncrement`	1	The amount to increment capacity when the initial attempt was successful.
`maxCapacity`	500	The maximum capacity of the token bucket. The number of available tokens cannot exceed this number.
`refillUnitsPerSecond`	0	The amount of capacity re-added to the bucket every second. A value of 0 means that no capacity is automatically re-added. (For example, only successful attempts result in incrementing capacity). A value of 0 requires `useCircuitBreakerMode` to be TRUE.
`retryCost`	5	The amount to decrement from the bucket for an attempt following a transient failure. The same amount is re-incremented back to

Parameter	Default value	Description
		the bucket if the attempt is successful.
`timeoutRetryCost`	10	The amount to decrement from the bucket for an attempt following a timeout or throttling failure. The same amount is re-incremented back to the bucket if the attempt is successful.
`useCircuitBreakerMode`	TRUE	Determines the behavior when an attempt to decrement capacity would result in the bucket's capacity to fall below zero. When TRUE, the token bucket will throw an exception indicating that no more retry capacity exists. When FALSE, the token bucket will delay the attempt until sufficient capacity has refilled.

For detailed information about exception types thrown during retry scenarios, including circuit breaker exceptions, see the section called “What exceptions reach your code when retries fail”.

Configure adaptive retries

As an alternative to the standard retry strategy, the adaptive retry strategy is an advanced approach that seeks the ideal request rate to minimize throttling errors.

Important

Adaptive retries is an advanced retry mode. Using this retry strategy is not normally recommended.

Adaptive retries includes all the features of standard retries. It adds a client-side rate limiter that measures the rate of throttled requests compared to non-throttled requests. It also limits traffic to attempt to stay within a safe bandwidth, ideally causing zero throttling errors.

The rate adapts in real time to changing service conditions and traffic patterns and might increase or decrease the rate of traffic accordingly. Critically, the rate limiter might delay initial attempts in high-traffic scenarios.

You select the adaptive retry strategy by providing an additional parameter to the `retryStrategy` method. The rate limiter parameters are configurable in the `rateLimiter` DSL block.

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

Note

The adaptive retry strategy assumes that the client works against a single resource (for example, one DynamoDB table or one Amazon S3 bucket).
If you use a single client for multiple resources, throttling or outages associated with one resource result in increased latency and failures when the client accesses all other resources. When you use the adaptive retry strategy, we recommend that you use a single client for each resource.

Observability

Observability is the extent to which a system's current state can be inferred from the data it emits. The data emitted is commonly referred to as telemetry.

The AWS SDK for Kotlin can provide all three common telemetry signals: metrics, traces, and logs. You can wire up a `TelemetryProvider` to send telemetry data to an observability backend (such as AWS X-Ray or Amazon CloudWatch) and then act on it.

By default, only logging is enabled and other telemetry signals are disabled in the SDK. This topic explains how to enable and configure telemetry output.

> **Important**
>
> `TelemetryProvider` is currently an experimental API that must be opted in to use.

Configure a `TelemetryProvider`

You can configure a `TelemetryProvider` in your application globally for all service clients or for individual clients. The following examples use a hypothetical `getConfiguredProvider()` function to demonstrate the `TelemetryProvider` API operations. The the section called "Telemetry providers" section describes information for implementations provided by the SDK. If a provider isn't supported, you can implement your own support or open a feature request on GitHub.

Configure the default global telemetry provider

By default, every service client attempts to use the globally available telemetry provider. This way, you can set the provider once, and all clients will use it. This should be done only once, before you instantiate any service clients.

To use the global telemetry provider, first update your project dependencies to add the telemetry defaults module as shown in the following Gradle snippet.

(You can navigate to the *X.Y.Z* link to see the latest version available.)

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

```
    ...
}
```

Then set the global telemetry provider before creating a service client as shown in the following code.

```
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 a telemetry provider for a specific service client

You can configure an individual service client with a specific telemetry provider (other than the global one). This is shown in the following example.

```
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")
}
```

Metrics

The following table lists the telemetry metrics that the SDK emits. Configure a telemetry provider to make the metrics observable.

What metrics are emitted?

Metric name	Units	Type	Attributes	Description
smithy.client.call.duration	s	Histogra	rpc.service, rpc.method	Overall call duration (including retries)
smithy.client.call.attempts	{attempt	Monotor Counter	rpc.service, rpc.method	The number of attempts for an individual operation
smithy.client.call.errors	{error}	Monotor Counter	rpc.service, rpc.method, exception.type	The number of errors for an operation
smithy.client.call.attempt_duration	s	Histogra	rpc.service, rpc.method	The time it takes to connect to the service, send the request, and get back HTTP status code and headers (including time queued waiting to be sent)
smithy.client.call.resolve_endpoint_duration	s	Histogra	rpc.service, rpc.method	The time it takes to resolve an endpoint (endpoint resolver, not DNS) for the request
smithy.client.call.serialization_duration	s	Histogra	rpc.service, rpc.method	The time it takes to serialize a message body
smithy.client.call.deserialization_duration	s	Histogra	rpc.service, rpc.method	The time it takes to deserialize a message body

Metric name	Units	Type	Attributes	Description
smithy.client.call.auth.signing_duration	s	Histogra	rpc.service, rpc.method, auth.scheme_id	The time it takes to sign a request
smithy.client.call.auth.resolve_identity_duration	s	Histogra	rpc.service, rpc.method, auth.scheme_id	The time it takes to acquire an identity (such as AWS credentials or a bearer token) from an Identity Provider
smithy.client.http.connections.acquire_duration	s	Histogra		The time it takes a request to acquire a connection
smithy.client.http.connections.limit	{connection}	[Async]UDownCoer		The maximum open connections allowed/configured for the HTTP client
smithy.client.http.connections.usage	{connection}	[Async]UDownCoer	state: idle | acquired	Current state of connections pool
smithy.client.http.connections.uptime	s	Histogra		The amount of time a connection has been open
smithy.client.http.requests.usage	{request	[Async]UDownCoer	state: queued | in-flight	The current state of HTTP client request concurrency

Metric name	Units	Type	Attributes	Description
smithy.client.http.requests.queued_duration	s	Histogra		The amount of time a request spent queued and waiting to be executed by the HTTP client
smithy.client.http.bytes_sent	By	Monotor Counter	server.address	The total number of bytes sent by the HTTP client
smithy.client.http.bytes_received	By	Monotor Counter	server.address	The total number of bytes received by the HTTP client

Following are the column descriptions:

- **Metric name**–The name of the emitted metric.
- **Units**–The unit of measure for the metric. Units are given in the UCUM case sensitive ("c/s") notation.
- **Type**–The type of instrument used to capture the metric.
- **Description**–A description of what the metric is measuring.
- **Attributes**–The set of attributes (dimensions) emitted with the metric.

Logging

The AWS SDK for Kotlin configures an SLF4J compatible logger as the default `LoggerProvider` of the telemetry provider. With SLF4J, which is an abstraction layer, you can use of any one of several logging systems at runtime. Supported logging systems include the Java Logging APIs, Log4j 2, and Logback.

> **Warning**
>
> We recommend that you only use wire logging for debugging purposes. (Wire logging is discussed below.) Turn it off in your production environments because it can log sensitive

data such as email addresses, security tokens, API keys, passwords, and AWS Secrets Manager secrets. Wire logging logs the full request or response without encryption, even for an HTTPS call.
For large requests (such as uploading a file to Amazon S3) or responses, verbose wire logging can also significantly impact your application's performance.

Example Log4j 2 logging configuration

While any SLF4J-compatible log library may be used, this example enables log output from the SDK in JVM programs using Log4j 2:

Gradle dependencies

(You can navigate to the *X.Y.Z* link to see the latest version available.)

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

Log4j 2 configuration file

Create a file named `log4j2.xml` in your `resources` directory (for example, `<project-dir>/src/main/resources`). Add the following XML configuration to the file:

```
<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>
```

This configuration includes the `%X` specifier in the `pattern` attribute that enables MDC (mapped diagnostic context) logging.

The SDK adds the following MDC elements for each operation.

rpc

The name of the invoked RPC, for example `S3.GetObject`.

sdkInvocationId

A unique ID assigned by the service client for the operation. The ID correlates all logging events related to the invocation of a single operation.

Specify log mode for wire-level messages

By default, the AWS SDK for Kotlin doesn't log wire-level messages because they might contain sensitive data from API requests and responses. However, sometimes you need this level of detail for debugging purposes.

With the Kotlin SDK, you can set a log mode in code or using environment settings to enable debug messaging for the following:

- HTTP requests
- HTTP responses

The log mode is backed by a bit-field where each bit is a flag (mode) and values are additive. You can combine one request mode and one response mode.

Set log mode in code

To opt into additional logging, set the `logMode` property when you construct a service client.

The following example shows how to enable logging of requests (with the body) and the response (without the body).

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

// ...

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

A log mode value set during service client construction, overrides any log mode value set from the environment.

Set log mode from the environment

To set a log mode globally for all service clients not explicitly configured in code, use one of the following:

- JVM system property: `sdk.logMode`
- Environment variable: `SDK_LOG_MODE`

The following case-insensitive values are available:

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

To create a combined log mode using settings from the environment, you separate the values with a pipe (`|`) symbol.

For example, the following examples set the same log mode as the previous example.

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

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

> **Note**
>
> You must also configure a compatible SLF4J logger and set the logging level to DEBUG to enable wire-level logging.

Telemetry providers

The SDK currently supports OpenTelemetry (OTel) as a provider. The SDK might offer additional telemetry providers in the future.

Topics

- Configure the OpenTelemetry-based telemetry provider

Configure the OpenTelemetry-based telemetry provider

The SDK for Kotlin provides an implementation of the `TelemetryProvider` interface backed by OpenTelemetry.

Prerequisites

Update your project dependencies to add the OpenTelemetry provider as shown in the following Gradle snippet. You can navigate to the *X.Y.Z* link to see the latest version available.

```
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")
}
```

Configure the SDK

The following code configures a service client by using the OpenTelemetry telemetry provider.

```
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**
>
> A discussion of how to configure the OpenTelemetry SDK is outside of the scope of this guide. The OpenTelemetry Java documentation contains configuration information on the various approaches: manually, automatically through the Java agent, or the (optional) collector.

Resources

The following resources are available to help you get started with OpenTelemetry.

- AWS Distro for OpenTelemetry - AWS OTeL Distro homepage
- aws-otel-java-instrumentation - AWS Distro for OpenTelemetry Java Instrumentation Library
- aws-otel-lambda - AWS managed OpenTelemetry Lambda layers
- aws-otel-collector - AWS Distro for OpenTelemetry Collector
- AWS Observability Best Practices - General best practices for observability specific to AWS

Override service client configuration

After a service client is created, the service client uses a fixed configuration for all operations. However, sometimes you might need to override the configuration for one or more specific operations.

Each service client has a `withConfig` extension so that you can modify a copy of the existing configuration. The `withConfig` extension returns a new service client with a modified configuration. The original client exists independently and uses its original configuration.

The following example shows the creation of an `S3Client` instance that calls two operations.

```
val s3 = S3Client.fromEnvironment {
```

```
    logMode = LogMode.LogRequest
    region = "us-west-2"
    // ...other configuration settings...
}

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

The following snippet shows how to override the configuration for a single `listObjectV2` operation.

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

The operation calls on the `s3` client use the original configuration that was specified when the client was created. Its configuration includes request logging and `us-west-2 region` for the Region.

The `listObjectsV2` invocation on the `overriddenS3` client uses same settings as the original `s3` client except for the Region, which is now `eu-central-1`.

Lifecycle of an overridden client

In the previous example, the `s3` client and the `overriddenS3` client are independent of each other. Operations can be invoked on either client for as long as they remain open. Each uses a separate configuration, but they can share underlying resources (such as an HTTP engine) unless those are also overridden.

You close a client with an overridden configuration and the original client separately. You can close a client with overridden configuration before or after you close its original client. Unless you need to use a client with overridden configuration for a long time, we recommend that you wrap its lifecycle with the `use` method. The `use` method ensures that the client is closed if exceptions occur.

Resources shared between clients

When you create a service client by using `withConfig`, it might share resources with the original client. In contrast, when you create a client by using fromEnvironment or you explicitly configure it,

the client uses independent resources. Resources such as HTTP engines and credentials providers are shared unless they are overridden in the `withConfig` block.

Because the lifecycle of each client is independent, shared resources remain open and usable until the last client is closed. Therefore, it is important for you to close overridden service clients when you no longer need them. This prevents shared resources from remaining open and consuming system resources such as memory, connection, and CPU cycles.

The following example shows both shared and independent resources.

The `s3` and `overriddenS3` clients share the same credentials provider instance, including its caching configuration. Calls made by `overriddenS3` reuse credentials if the cached value is still current from calls made by the `s3` client.

The HTTP engine is not shared between the two clients. Each client has an independent HTTP engine because it was overridden in the `withConfig` call.

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

s3.listBuckets { ... }

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

Use the SDK

This section provides basic information required to use the AWS SDK for Kotlin.

Topics

- Make requests
- Coroutines
- Streaming operations
- Pagination
- Waiters
- Error handling
- Presign requests
- Troubleshooting FAQs
- Mocking in the AWS SDK for Kotlin

Make requests

Use a service client to make requests to an AWS service. The AWS SDK for Kotlin provides Domain Specific Languages (DSLs) following a type-safe builder pattern to create requests. Nested structures of requests are also accessible through their DSLs.

The following example shows how to create an Amazon DynamoDB createTable operation input:

```
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)
```

Service interface DSL overloads

Each non-streaming operation on the service client interface has a DSL overload so that you don't have to create a separate request.

Example of creating an Amazon Simple Storage Service (Amazon S3) bucket with the overloaded function:

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

This is equivalent to:

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

```
s3client.createBucket(request)
```

Requests with no required inputs

Operations that don't have required inputs can be called without having to pass a request object. This is often possible with list-type operations, such as the Amazon S3 `listBuckets` API operation.

For example, the following three statements are equivalent:

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

Coroutines

The AWS SDK for Kotlin is asynchronous by default. The SDK for Kotlin uses `suspend` functions for all operations, which are meant to be called from a coroutine.

For a more in-depth guide to coroutines, see the official Kotlin documentation.

Making concurrent requests

The async coroutine builder can be used to launch concurrent requests where you care about the results. `async` returns a Deferred, which represents a light-weight, non-blocking future that represents a promise to provide a result later.

If you don't care about the results (only that an operation completed), you can use the launch coroutine builder. `launch` is conceptually similar to `async`. The difference is that launch returns a Job and does not carry any resulting value, while `async` returns a `Deferred`.

The following is an example of making concurrent requests to Amazon S3 using the headObject operation to get the content size of two keys:

```
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")

}
```

Making blocking requests

To make service calls from existing code that doesn't use coroutines and implements a different threading model, you can use the runBlocking coroutine builder. An example of a different threading model is using Java's traditional executors/futures approach. You might need to use this approach if you're blending Java and Kotlin code or libraries.

As its name suggests, this `runBlocking` builder launches a new coroutine and blocks the current thread until it completes.

> **Warning**
>
> `runBlocking` should not generally be used from a coroutine. It is designed to bridge regular blocking code to libraries that are written in suspending style (such as in main functions and tests).

Streaming operations

In the AWS SDK for Kotlin, binary data (streams) are represented as a `ByteStream` type, which is an abstract read-only stream of bytes.

Streaming responses

Responses with a binary stream (such as the Amazon Simple Storage Service (Amazon S3) GetObject API operation) are handled differently from other methods. These methods take a lambda function that handles the response rather than returning the response directly. This limits the scope of the response to the function and simplifies lifetime management for both the caller and the SDK runtime.

After the lambda function returns, any resources like the underlying HTTP connection are released. (The `ByteStream` should not be accessed after the lambda returns and should not be passed out of the closure.) The result of the call is whatever the lambda returns.

The following code example shows the getObject function receiving a lambda parameter, which handles the response.

```
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")
```

The `ByteStream` type has the following extensions for common ways of consuming it:

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

All of these are defined in the `aws.smithy.kotlin.runtime.content` package.

Streaming requests

To supply a `ByteStream`, there are also several convenience methods, including the following:

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

All of these are defined in the `aws.smithy.kotlin.runtime.content` package.

The following code example shows the use of `ByteStream` convenience methods that provide the body property in the creation of a PutObjectRequest :

```
val req = PutObjectRequest {
    ...
    body = ByteStream.fromFile(file)
    // body = ByteStream.fromBytes(byteArray)
    // body = ByteStream.fromString("string")
```

```
    // etc
}
```

Pagination

Many AWS operations return paginated results when the payload is too large to return in a single response. The AWS SDK for Kotlin includes extensions to the service client interface that auto paginate the results for you. You only have to write the code that processes the results.

Pagination is exposed as a Flow<T> so that you can take advantage of Kotlin's idiomatic transforms for asynchronous collections (such as `map`, `filter`, and `take`). Exceptions are transparent, which makes error handling feel like a regular API call, and cancellation adheres to the general cooperative cancellation of coroutines. For more information, see flows and flow exceptions in the official guide.

> **Note**
>
> The following examples use Amazon S3. However, the concepts are the same for any service that has one or more paginated APIs. All pagination extensions are defined in the `aws.sdk.kotlin.<service>.paginators` package (such as `aws.sdk.kotlin.dynamodb.paginators`).

The following code example shows how you can process the paginated response from the listObjectsV2Paginated function call.

Imports

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

Code

```
val s3 = S3Client.fromEnvironment()
val req = ListObjectsV2Request {
    bucket = "amzn-s3-demo-bucket"
    maxKeys = 1
}
```

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

Waiters

Waiters are a client-side abstraction used to poll a resource until a desired state is reached, or until it is determined that the resource will not enter the desired state. This is a common task when working with services that are eventually consistent, like Amazon Simple Storage Service (Amazon S3), or services that asynchronously create resources, like Amazon EC2.

Writing logic to continuously poll the status of a resource can be cumbersome and error-prone. The goal of waiters is to move this responsibility out of customer code and into the AWS SDK for Kotlin, which has in-depth knowledge of the timing aspects for the AWS operation.

Note

The following examples use Amazon S3. However, the concepts are the same for any AWS service that has one or more waiters defined. All extensions are defined in the `aws.sdk.kotlin.<service>.waiters` package (such as `aws.sdk.kotlin.dynamodb.waiters`). They also follow a standard naming convention (`waitUntil<Condition>`).

The following code example shows the use of a waiter function that allows you to avoid writing polling logic.

Imports

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

Code

```
val s3 = S3Client.fromEnvironment()
```

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

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

// The bucket now exists.
```

Note

Each wait method returns an `Outcome` instance that can be used to get at the final response that corresponds to reaching the desired condition. An outcome also contains additional details like the number of attempts made to reach the desired state.

Error handling

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

Service exceptions

The most common exception is `AwsServiceException`, from which all service-specific exceptions (such as `S3Exception`) inherit. This exception represents an error response from an AWS service. For example, if you try to terminate an Amazon EC2 instance that doesn't exist, Amazon EC2 returns an error response. The error response details are included in the `AwsServiceException` that's thrown.

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

Client exceptions

`ClientException` indicates that a problem occurred inside the AWS SDK for Kotlin client code, either while trying to send a request to AWS or while trying to parse a response from AWS. A

`ClientException` is generally more severe than an `AwsServiceException` and indicates that a major problem is preventing the client from processing service calls to AWS services. For example, the AWS SDK for Kotlin throws a `ClientException` if it fails to parse a response from a service.

Error metadata

Every service exception and client exception has the `sdkErrorMetadata` property. This is a typed property bag that can be used to retrieve additional details about the error.

Several predefined extensions exist to the `AwsErrorMetadata` type directly, including but not limited to the following:

- `sdkErrorMetadata.requestId` – the unique request id
- `sdkErrorMetadata.errorMessage` – the human readable message (usually matches the `Exception.message`, but might contain more information if the exception was unknown to the service)
- `sdkErrorMetadata.protocolResponse` – The raw protocol response

The following example demonstrates accessing the error metadata.

```
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}")
}
```

Presign requests

You can presign requests for some AWS API operations so that another caller can use the request later without presenting their own credentials.

For example, assume that Alice has access to an Amazon Simple Storage Service (Amazon S3) object and she wants to temporarily share object access with Bob. Alice can generate a presigned `GetObject` request to share with Bob so that he can download the object without requiring access to Alice's credentials.

Presigning basics

The SDK for Kotlin provides extension methods on service clients to presign requests. All presigned requests require a duration that represents how long the signed request is valid. After the duration ends, the presigned request expires and raises an authentication error if executed.

The following code shows an example that creates a presigned `GetObject` request for Amazon S3. The request is valid for 24 hours after creation.

```
val s3 = S3Client.fromEnvironment()

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

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

The `presignGetObject` extension method returns an `HttpRequest` object. The request object contains the presigned URL where the operation can be invoked. Another caller can use the URL (or the entire request) in a different codebase or programming language environment.

After creating the presigned request, use an HTTP client to invoke a request. The API to invoke an HTTP GET request depends on the HTTP client. The following example uses the Kotlin `URL.readText` method.

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

Advanced presigning configuration

In the SDK, each method that can presign requests has an overload that you can use to provide advanced configuration options, such as a specific signer implementation or detailed signing parameters.

The following example shows an Amazon S3 `GetObject` request that uses the CRT signer variant and specifies a future signing date.

```
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
}
```

The returned presigned request is forward-dated 24 hours and is not valid before then. It expires 8 hours after that.

Presigning POST and PUT requests

Many operations that are presignable require only a URL and must be executed as HTTP GET requests. Some operations, however, take a body and must be executed as an HTTP POST or HTTP PUT request along with headers in some cases. Presigning these requests is identical to presigning GET requests, but invoking the presigned request is more complicated.

Here is an example of presigning an S3 `PutObject` request:

```
val s3 = S3Client.fromEnvironment()

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

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

The returned `HttpRequest` has a method value of `HttpMethod.PUT` and includes a URL and headers that must be included in the future invocation of the HTTP request. You can pass this request to a caller that can execute it in a different codebase or programming language environment.

After creating the presigned POST or PUT request, use an HTTP client to invoke a request. The API to invoke a POST or PUT request URL depends on the HTTP client used. The following example uses an OkHttp HTTP client and includes a body that contains `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()
```

Operations the SDK can presign

The SDK for Kotlin currently supports presigning the following API operations that need to be called with the associated HTTP method.

AWS service	Operation	SDK extension method	Use HTTP method
Amazon S3	GetObject	presignGetObject	HTTP GET
Amazon S3	PutObject	presignPutObject	HTTP PUT
Amazon S3	UploadPart	presignUploadPart	HTTP PUT
AWS Security Token Service	GetCallerIdentity	presignGetCallerIdentity	HTTP POST
Amazon Polly	SynthesizeSpeech	presignSynthesizeSpeech	HTTP POST

Troubleshooting FAQs

As you use the AWS SDK for Kotlin in your applications, you might encounter some of the issues listed in this topic. Use the following suggestions to help uncover the root cause and resolve the error.

How do I fix "connection closed" issues?

You might encounter “connection closed” issues as exceptions such as one of the following types:

- `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)`

These exceptions indicate that a TCP connection from the SDK to a service was unexpectedly closed or reset. The connection might have been closed by your host, the AWS service, or an intermediary party such as a NAT gateway, proxy, or load balancer.

These types of exceptions are automatically retried but might still appear in SDK logs, depending on your logging configuration. If the exception is thrown into your code, that indicates the active retry strategy has exhausted its configured limits such as maximum attempts or retry token bucket. See the the section called “Retries” section of this guide for more information about retry strategies. See also the section called “Why are exceptions thrown before reaching the maximum attempts?”.

Why are exceptions thrown before reaching the maximum attempts?

Sometimes you might see exceptions that you expected to be retried but were thrown instead. In these situations, the following steps might help resolve the issue.

- **Verify that the exception is retryable.** Some exceptions are not retryable, such as those that indicate a malformed service request, lack of permissions, and non-existent resources, as examples. The SDK does not automatically retry these kinds of exceptions. For information on how to check for retryable exceptions, see the section called “Check if an exception is retryable”.
- **Verify that the exception is being thrown into your code.** Some exceptions appear in log messages as information but are not actually thrown into your code. For instance, retryable exceptions such as throttling errors might be logged as the SDK automatically works through multiple backoff-and-retry cycles. The invocation of an SDK operation throws an exception only if it was not handled by the configured retry settings.
- **Verify your configured retry settings.** See the the section called “Retries” section of this guide for more information about retry strategies and retry policies. Ensure that your code is using the settings you expect or the automatic defaults.
- **Consider adjusting your retry settings.** After you verify the previous items, but the issue is not resolved, you might consider adjusting retry settings.

- **Increase the maximum number of attempts.** By default the maximum number of attempts for an operation is 3. If you find that this is not enough and exceptions are still occurring at the default setting, consider increasing the `retryStrategy.maxAttempts` property in your client configuration. See the section called "Maximum attempts" for more information.
- **Increase the delay settings.** Some exceptions might be retried too rapidly before the underlying condition has had a chance to resolve. If you suspect that do be the case, consider increasing the `retryStrategy.delayProvider.initialDelay` or `retryStrategy.delayProvider.maxBackoff` properties in your client configuration. See the section called "Delays and backoff" for more information.
- **Disable circuit breaker mode.** The SDK maintains a bucket of tokens for each service client by default. When the SDK attempts a request and it fails with a retryable exception, the token count is decremented; when the request succeeds the token count is incremented.

  By default, if this token bucket reaches 0 tokens remaining, the circuit is broken. After the circuit is broken, the SDK disables retries and any current and subsequent requests that fail on the first attempt immediately throw an exception. The SDK re-enables retries after successful initial attempts return enough capacity to the token bucket. This behavior is intentional and designed to prevent retry storms during service outages and service recovery.

  If you prefer that the SDK continue retrying up to the maximum configured attempts, consider disabling circuit breaker mode by setting the `retryStrategy.tokenBucket.useCircuitBreakerMode` property to false in your client configuration. With this property set to false, the SDK client waits until the token bucket reaches sufficient capacity rather than abandon further retries that might lead to an exception when there are 0 tokens remaining.

How do I fix `NoSuchMethodError` or NoClassDefFoundError?

These errors are most commonly caused by missing or conflicting dependencies. See the section called "How do I resolve dependency conflicts?" for more information.

I see a `NoClassDefFoundError` for `okhttp3/coroutines/ExecuteAsyncKt`

This indicates a dependency problem for OkHttp specifically. See the section called "Resolving OkHttp version conflicts in your application" for more information.

How do I resolve dependency conflicts?

When you use the AWS SDK for Kotlin, it needs certain AWS and third-party dependencies to work properly. If these dependencies are missing or unexpected versions at runtime, you might see errors like `NoSuchMethodError` or `NoClassDefFoundError`. These dependency issues usually fall into two groups:

- SDK/Smithy dependency conflicts
- Third-party dependency conflicts

When you build your Kotlin application, you'll likely use Gradle to manage dependencies. Adding a dependency on an SDK service client to your project automatically includes all necessary related dependencies. However, if your application has other dependencies, they might clash with those required by the SDK. For example, the SDK relies on OkHttp, a popular HTTP client that your application might also use. To help you spot these conflicts, Gradle offers a handy task the lists your project's dependencies:

```
./gradlew dependencies
```

When you encounter dependency conflicts, you might need to take action. You can either specify a particular version of a dependency or shadow dependencies into a local namespace. Gradle dependency resolution is a complex topic that is discussed in the following sections of the *Gradle User Manual*:

- Understanding dependency resolution
- Dependency constraints and conflict resolution
- Aligning dependency versions

Managing SDK and Smithy dependencies in your project

When you use the SDK, keep in mind that its modules typically depend on other SDK modules with matching version numbers. For example, `aws.sdk.kotlin:s3:1.2.3` depends on `aws.sdk.kotlin:aws-http:1.2.3`, which depends on `aws.sdk.kotlin:aws-core:1.2.3`, and so on.

The SDK modules also use specific Smithy module versions. While Smithy module versions don't sync with SDK version numbers, they must match the SDK's expected version. For example,

`aws.sdk.kotlin:s3:1.2.3` might depend on `aws.smithy.kotlin:serde:1.1.1`, which depends on `aws.smithy.kotlin:runtime-core:1.1.1`, and so on.

To avoid dependency conflicts, upgrade all your SDK dependencies together, and do the same for any explicit Smithy dependencies. Consider using our Gradle version catalog to keep versions in sync and eliminate guesswork in mapping between SDK and Smithy versions.

Remember that minor version updates in SDK/Smithy modules may include breaking changes, as outlined in our versioning policy. When upgrading between minor versions, carefully review changelogs and thoroughly test runtime behavior.

Resolving OkHttp version conflicts in your application

OkHttp is a popular HTTP engine that the SDK uses by default on JVM. Your application might include other dependencies or frameworks that bring in a different OkHttp version. This can cause a `NoClassDefFoundError` for classes in the `okhttp3` namespace, such as `okhttp/coroutines/ExecuteAsyncKt` or `okhttp3/ConnectionListener`. When this happens, you should generally choose the newer version to resolve conflicts. To help you trace the sources of these conflicts, Gradle offers a useful task. You can list all dependencies by running:

```
./gradlew dependencies
```

For instance, if the SDK depends on OkHttp `5.0.0-alpha.14` and another dependency such as Spring Boot depends on OkHttp `4.12.0` then you should use the `5.0.0-alpha.14 version`. You can do this with a `constraints` block in Gradle:

```
dependencies {
    constraints {
        implementation("com.squareup.okhttp3:okhttp:4.12.0")
    }
}
```

Alternatively, if you must use OkHttp 4.x, the SDK provides an `OkHttp4Engine`. See the documentation for information on how to configure Gradle and use `OkHttp4Engine` in your code.

Mocking in the AWS SDK for Kotlin

Developers can use several frameworks to perform mocking in tests with the AWS SDK for Kotlin. This topic documents extra configuration or special considerations that some frameworks require.

MockK

When you mock module-wide extension functions using MockK, you need to do extra configuration. In the SDK for Kotlin, paginators, waiters, and presigners are examples of extension functions, so when mocking their behavior, you need extra configuration.

You must call `mockkStatic("<MODULE_CLASS_NAME>")` before setting up your mocks. As a general rule, the module class names are:

- **Paginators**: `aws.sdk.kotlin.services.<service>.paginators.PaginatorsKt`
- **Waiters**: `aws.sdk.kotlin.services.<service>.waiters.WaitersKt`
- **Presigners**: `aws.sdk.kotlin.services.<service>.presigners.PresignersKt`

For example, in the following test that includes mocking `listBucketsPaginated`—a paginator extenstion function—we add `mockkStatic("aws.sdk.kotlin.services.s3.paginators.PaginatorsKt")`:

```
    @Test
    fun testPaginatedListBuckets() = runTest {

        mockkStatic("aws.sdk.kotlin.services.s3.paginators.PaginatorsKt")
        val s3Client: S3Client = mockk()
        val s3BucketLister = S3BucketLister(s3Client)

        val expectedBuckets = listOf(
            Bucket { name = "bucket1" },
            Bucket { name = "bucket2" }
        )

        val response = ListBucketsResponse { buckets = expectedBuckets }
        coEvery { s3Client.listBucketsPaginated() } returns flowOf(response)

        val result = s3BucketLister.getAllBucketNames()

        assertEquals(listOf("bucket1", "bucket2"), result)
    }
```

Without `mockkStatic`, you see the following error:

```
Missing mocked calls inside every { ... } block: make sure the object inside the block
 is a mock
io.mockk.MockKException: Missing mocked calls inside every { ... } block: make sure the
 object inside the block is a mock
    at
 io.mockk.impl.recording.states.StubbingState.checkMissingCalls(StubbingState.kt:14)
    at io.mockk.impl.recording.states.StubbingState.recordingDone(StubbingState.kt:8)
    at io.mockk.impl.recording.CommonCallRecorder.done(CommonCallRecorder.kt:47)
    at io.mockk.impl.eval.RecordedBlockEvaluator.record(RecordedBlockEvaluator.kt:63)
    at io.mockk.impl.eval.EveryBlockEvaluator.every(EveryBlockEvaluator.kt:30)
    at io.mockk.MockKDsl.internalCoEvery(API.kt:100)
    at io.mockk.MockKKt.coEvery(MockK.kt:174)
```

In the case of a presigner extension function without `mockkStatic`, you might see:

```
key is bound to the URI and must not be null
java.lang.IllegalArgumentException: key is bound to the URI and must not be null
    at
 aws.sdk.kotlin.services.s3.serde.GetObjectOperationSerializer.serialize(GetObjectOperationSeri
    at
 aws.sdk.kotlin.services.s3.presigners.PresignersKt.presignGetObject(Presigners.kt:49)
    at aws.sdk.kotlin.services.s3.presigners.PresignersKt.presignGetObject
$default(Presigners.kt:40)
    at aws.sdk.kotlin.services.s3.presigners.PresignersKt.presignGetObject-
exY8QGI(Presigners.kt:30)
```

All example artifacts

Code under test

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.paginators.listBucketsPaginated
import kotlinx.coroutines.flow.filter
import kotlinx.coroutines.flow.toList
import kotlinx.coroutines.flow.transform
import kotlinx.coroutines.runBlocking
import org.slf4j.Logger
import org.slf4j.LoggerFactory

fun main() {
    val logger: Logger = LoggerFactory.getLogger(::main.javaClass)
```

```
    // Create an S3Client
    S3Client { region = "us-east-1" }.use { s3Client ->
        // Create service instance
        val bucketLister = S3BucketLister(s3Client)
        // Since getAllBucketNames is a suspend function, you'll need to run it in a
 coroutine scope
        runBlocking {
            val bucketNames = bucketLister.getAllBucketNames()
            logger.info("Found buckets: $bucketNames")
        }
    }
}

class S3BucketLister(private val s3Client: S3Client) {
    suspend fun getAllBucketNames(): List<String> {
        return s3Client.listBucketsPaginated()
            .transform { response ->
                response.buckets?.forEach { bucket ->
                    emit(bucket.name ?: "")
                }
            }
            .filter { it.isNotEmpty() }
            .toList()
    }
}
```

Test class

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.model.Bucket
import aws.sdk.kotlin.services.s3.model.ListBucketsResponse
import aws.sdk.kotlin.services.s3.paginators.listBucketsPaginated
import io.mockk.coEvery
import io.mockk.mockk
import io.mockk.mockkStatic
import kotlinx.coroutines.flow.flowOf
import kotlinx.coroutines.test.runTest
import org.junit.jupiter.api.Assertions.assertEquals
import org.junit.jupiter.api.Test

class S3BucketListerTest {

    @Test
```

```
    fun testPaginatedListBuckets() = runTest {

        mockkStatic("aws.sdk.kotlin.services.s3.paginators.PaginatorsKt")
        val s3Client: S3Client = mockk()
        val s3BucketLister = S3BucketLister(s3Client)

        val expectedBuckets = listOf(
            Bucket { name = "bucket1" },
            Bucket { name = "bucket2" }
        )

        val response = ListBucketsResponse { buckets = expectedBuckets }
        coEvery { s3Client.listBucketsPaginated() } returns flowOf(response)

        val result = s3BucketLister.getAllBucketNames()

        assertEquals(listOf("bucket1", "bucket2"), result)
    }
}
```

build.gradle.kts

```
plugins {
    kotlin("jvm") version "2.1.20"
    application
}

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

repositories {
    mavenCentral()
}

dependencies {
    implementation(platform(awssdk.bom))
    implementation(platform("org.apache.logging.log4j:log4j-bom:2.24.3"))

    implementation(awssdk.services.s3)
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")

    // Testing Dependencies
    testImplementation(platform("org.junit:junit-bom:5.11.0"))
```

```
    testImplementation("org.junit.jupiter:junit-jupiter")
    testImplementation("org.jetbrains.kotlinx:kotlinx-coroutines-test:1.7.3")
    testImplementation("io.mockk:mockk:1.14.0")
}

tasks.test {
    useJUnitPlatform()
}

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

application {
    mainClass = "org.example.S3BucketService"
}
```

settings.gradle.kts

```
plugins {
    id("org.gradle.toolchains.foojay-resolver-convention") version "0.10.0"
}
rootProject.name = "mockK-static"

dependencyResolutionManagement {
    repositories {
        mavenCentral()
    }

    versionCatalogs {
        create("awssdk") {
            from("aws.sdk.kotlin:version-catalog:1.4.69")
        }
    }
}
```

Work with AWS services using the AWS SDK for Kotlin

This chapter contains information about how to work with AWS services by using the SDK for Kotlin.

Contents

- Work with Amazon S3 using the AWS SDK for Kotlin
  - Data integrity protection with checksums
    - Upload an object
      - Use a pre-calculated checksum value
      - Multipart uploads
    - Download an object
      - Asynchronous validation
  - Work with Amazon S3 Multi-Region Access Points by using the SDK for Kotlin
    - Work with Multi-Region Access Points
    - Work with objects and Multi-Region Access Points
- Work with DynamoDB using the AWS SDK for Kotlin
  - Use AWS account-based endpoints
  - Map classes to DynamoDB items by using the DynamoDB Mapper (Developer Preview)
    - Get started with DynamoDB Mapper
      - Add dependencies
      - Create and use a mapper
      - Define a schema with class annotations
      - Invoke operations
        - Work with paginated responses
    - Configure DynamoDB Mapper
      - Use interceptors
        - Understand the request pipeline
        - Hooks
          - Read-only hooks
          - Modify hooks

        - Execution order
        - Example configuration
    - Generate a schema from annotations
        - Apply the plugin
        - Configure the plugin
        - Annotate classes
            - Class annotations
            - Property annotations
        - Define a custom item converter
    - Manually define schemas
        - Define a schema in code
    - Use secondary indices with DynamoDB Mapper
        - Define a schema for a secondary index
        - Use secondary indices in operations
    - Use expressions
        - Use expressions in operations
        - DSL components
            - Attributes
            - Equalities and inequalities
            - Ranges and sets
            - Boolean logic
            - Functions and properties
            - Sort key filters

Work with Amazon S3 using the AWS SDK for Kotlin

Your main interface to the Amazon Simple Storage Service for the Kotlin SDK is the S3Client. Use the `S3Client` like other service clients in the SDK to make requests to Amazon S3.

Resources to help you use the Kotlin SDK with S3 are:

- the Kotlin SDK API reference for S3.

- the S3 service User Guide and service API reference.

The following topics present guided code examples for select Kotlin SDK APIs that work with S3.

Topics

- Data integrity protection with checksums
- Work with Amazon S3 Multi-Region Access Points by using the SDK for Kotlin

Data integrity protection with checksums

Amazon Simple Storage Service (Amazon S3) provides the ability to specify a checksum when you upload an object. When you specify a checksum, it is stored with the object and can be validated when the object is downloaded.

Checksums provide an additional layer of data integrity when you transfer files. With checksums, you can verify data consistency by confirming that the received file matches the original file. For more information about checksums with Amazon S3, see the Amazon Simple Storage Service User Guide including the supported algorithms.

You have the flexibility to choose the algorithm that best fits your needs and let the SDK calculate the checksum. Alternatively, you can provide a pre-computed checksum value by using one of the supported algorithms.

> **Note**
>
> Beginning with version 1.4.0 of the AWS SDK for Kotlin, the SDK provides default integrity protections by automatically calculating a CRC32 checksum for uploads. The SDK calculates this checksum if you don't provide a precalculated checksum value or if you don't specify an algorithm that the SDK should use to calculate a checksum.
> The SDK also provides global settings for data integrity protections that you can set externally, which you can read about in the AWS SDKs and Tools Reference Guide.

We discuss checksums in two request phases: uploading an object and downloading an object.

Upload an object

You upload objects to Amazon S3 with the SDK for Kotlin by using the `putObject` function with a request parameter. The request data type provides the `checksumAlgorithm` property to enable checksum computation.

The following code snippet shows a request to upload an object with a CRC32 checksum. When the SDK sends the request, it calculates the CRC32 checksum and uploads the object. Amazon S3 stores the checksum with the object.

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

If you don't provide a checksum algorithm with the request, the checksum behavior varies depending on the version of the SDK that you use as shown in the following table.

Checksum behavior when no checksum algorithm is provided

Kotlin SDK version	Checksum behavior
earlier than 1.4.0	The SDK doesn't automatically calculate a CRC-based checksum and provide it in the request.
1.4.0 or later	The SDK uses the CRC32 algorithm to calculate the checksum and provides it in the request. Amazon S3 validates the integrity of the transfer by computing its own CRC32 checksum and compares it to the checksum provided by the SDK. If the checksums match, the checksum is saved with the object.

Use a pre-calculated checksum value

A pre-calculated checksum value provided with the request disables automatic computation by the SDK and uses the provided value instead.

The following example shows a request with a pre-calculated SHA256 checksum.

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

If Amazon S3 determines the checksum value is incorrect for the specified algorithm, the service returns an error response.

Multipart uploads

You can also use checksums with multipart uploads.

You must specify the checksum algorithm in the `CreateMultipartUpload` request and in each `UploadPart` request. As a final step, you must specify the checksum of each part in the `CompleteMultipartUpload`. The following example shows how to create a multipart upload with the checksum algorithm specified.

```
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
    }
}
```

Download an object

When you use the getObject method to download an object, the SDK automatically validates the checksum when the `checksumMode` property of the builder for the `GetObjectRequest` is set to `ChecksumMode.Enabled`.

The request in the following snippet directs the SDK to validate the checksum in the response by calculating the checksum and comparing the values.

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

> **Note**
>
> If the object wasn't uploaded with a checksum, no validation takes place.

If you use an SDK version of 1.4.0 or later, the SDK automatically checks the integrity of `getObject` requests without adding `checksumMode = ChecksumMode.Enabled` to the request.

Asynchronous validation

Because the SDK for Kotlin uses streaming responses when it downloads an object from Amazon S3, the checksum will be calculated as you consume the object. Therefore, you *must* consume the object so that the checksum is validated.

The following example shows how to validate a checksum by fully consuming the response.

```
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.
}
```

By contrast, the code in the following example doesn't use the object in any way, so the checksum is not validated.

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

If the checksum calculated by the SDK does not match the expected checksum sent with the response, the SDK throws a `ChecksumMismatchException`.

Work with Amazon S3 Multi-Region Access Points by using the SDK for Kotlin

Amazon S3 Multi-Region Access Points provide a global endpoint that applications can use to fulfill requests from Amazon S3 buckets located in multiple AWS Regions. You can use Multi-Region Access Points to build multi-Region applications with the same architecture used in a single Region, and then run those applications anywhere in the world.

The Amazon S3 User Guide contains more background information about Multi-Region Access Points.

Work with Multi-Region Access Points

To create a Multi-Region Access Point, start by specifying one bucket in each AWS Region that you want to serve requests. The following snippet creates two buckets.

Create buckets

The following function creates two buckets to work with the Multi-Region Access Point. One bucket is in Region `us-east-1` and the other is in Region `us-west-1`.

The creation of the S3 client that passed in as the first argument is shown in the first example under the section called “Work with objects”.

```
        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.")
        }
    }
```

You use the Kotlin SDK's S3 control client to create, delete, and get information about Multi-Region Access Points.

Add a dependency on the S3 control artifact as shown in the following snippet. (You can navigate to the *X.Y.Z* link to see the latest version available.)

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

Configure the S3 control client to work with AWS Region `us-west-2` as shown in the following code. All S3 control client operations must target the `us-west-2` region.

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

Use the S3 control client to create a Multi-Region Access Point by specifying the bucket names (created previously) as shown in the following code.

```
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"
    }
```

Because the creation of a Multi-Region Access Point is an asynchronous operation, you use the token that you receive from the immediate response to check on the status of the creation process. After the status check returns a success message, you can use the `GetMultiRegionAccessPoint` operation to get the Multi-Region Access Point's alias. The alias is the last component of the ARN, which you need for object-level operations.

Use token to check status

Use the `DescribeMultiRegionAccessPointOperation` to check the status of the last operation. After the `requestStatus` value becomes "SUCCEEDED", you can work with the Multi-Region Access Point.

```
        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)
            }
        }
```

Work with objects and Multi-Region Access Points

You use the S3 client to work with objects in Multi-Region Access Points. Many of the operations that you use on objects in buckets you can use on Multi-Region Access Points. For more information and a full list of operations, see Multi-Region Access Point compatibility with S3 operations.

Operations with Multi-Region Access Points are signed with the Asymmetric SigV4 (SigV4a) signing algorithm. To configure SigV4a, first add the following dependencies to your project. (You can navigate to the *X.Y.Z* link to see the latest version available.)

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

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

After you add the dependencies, configure the S3 client to use the SigV4a signing algorithm as shown in the following code.

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric SigV4 (SigV4a) signing
algorithm.
            val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4aScheme)
            }
            return s3
        }
```

After you configure the S3 client, operations that S3 supports for Multi-Region Access Points work the same. The only difference is that the bucket parameter must be the ARN of the Multi-Region Access Point. You can get the ARN from the Amazon S3 console or programmatically as shown previously in the `createMrap` function that returns an ARN.

The following code example shows the ARN used in a `GetObject` operation.

```
    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
    }
```

Work with DynamoDB using the AWS SDK for Kotlin

Use AWS account-based endpoints

DynamoDB offers AWS account-based endpoints that can improve performance by using your AWS account ID to streamline request routing.

To take advantage of this feature, you need to use version 1.3.37 or greater of the AWS SDK for Kotlin. You can find the latest version of the SDK listed in the Maven central repository. After a supported version of SDK is active, it automatically uses the new endpoints.

If you want to opt out of the account-based routing, you have four options:

- Configure a DynamoDB service client with the `AccountIdEndpointMode` set to `DISABLED`.
- Set an environment variable.
- Set a JVM system property.
- Update the shared AWS config file setting.

The following snippet is an example of how to disable account-based routing by configuring a DynamoDB service client:

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

The AWS SDKs and Tools Reference Guide provides more information on the last three configuration options.

Map classes to DynamoDB items by using the DynamoDB Mapper (Developer Preview)

> **DynamoDB Mapper is a Developer Preview release. It is not feature complete and is subject to change.**

DynamoDB Mapper is a high-level library that offers mechanisms to map Kotlin classes to DynamoDB tables and indices, similar to the AWS SDK for Java's DynamoDB Enhanced Client or the AWS SDK for .NET's Object Persistence Model.

You define schemas that describe your data object and how to convert them to DynamoDB items. After you define the schema, DynamoDB Mapper provides an intuitive interface to use your objects in create, read, update, or delete (CRUD) operations on your tables and indices.

Topics

- Get started with DynamoDB Mapper
- Configure DynamoDB Mapper
- Generate a schema from annotations
- Manually define schemas
- Use secondary indices with DynamoDB Mapper
- Use expressions

Get started with DynamoDB Mapper

> **DynamoDB Mapper is a Developer Preview release. It is not feature complete and is subject to change.**

The following tutorial introduces the basic components of DynamoDB Mapper and shows how to use it in your code.

Add dependencies

To begin working with DynamoDB Mapper in your Gradle project, add a plugin and two dependencies to your `build.gradle.kts` file.

(You can navigate to the *X.Y.Z* link to see the latest version available.)

```
// 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")
}
```

*Replace *<Version>* with the latest release of the SDK. To find the latest version of the SDK, check the latest release on GitHub.

> **Note**
>
> Some of these dependencies are optional if you plan to define schemas manually. See the section called “Manually define schemas” for more information and the reduced set of dependencies.

Create and use a mapper

DynamoDB Mapper uses the AWS SDK for Kotlin’s DynamoDB client to interact with DynamoDB. You need to provide a fully configured `DynamoDbClient` instance when you create a mapper instance as shown in the following code snippet:

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

val client = DynamoDbClient.fromEnvironment()
```

```
val mapper = DynamoDbMapper(client)
```

> **Note**
>
> `DynamoDbMapper` doesn't support table creation operations. Use the `DynamoDbClient` to create tables.

After you have created the mapper instance, you can use it to get the table instance as shown next:

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

The previous code gets a reference to a table in DynamoDB named `cars` with a schema defined by `CarSchema` (we discuss schemas below). After you create a table instance, you can perform operations against it. The following code snippet show two example operations against the `cars` table:

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

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

The previous code creates a new item in the `cars` table. The code creates a `Car` instance inline using the `Car` class, whose definition is shown below. Next, the code queries the `cars` table for items whose partition key is `Peugeot` and prints them. Operations are described in more detail below.

Define a schema with class annotations

For a variety of Kotlin classes, the SDK can automatically generate schemas at build time by using the DynamoDB Mapper Schema Generator plugin for Gradle. When you use the schema generator, the SDK inspects your classes to infer the schema, which alleviates some of the boilerplate involved in manually defining schemas. You can customize the schema that is generated by using additional annotations and configuration.

To generate a schema from annotations, first annotate your classes with @DynamoDbItem and any keys with @DynamoDbPartitionKey and @DynamoDbSortKey. The following code shows the annotated Car class:

```
// 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
)
```

After building, you can refer to the automatically generated CarSchema. You can use the reference in the mapper's getTable method to get a table instance as shown in the following:

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

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

Alternatively, you can get the table instance by taking advantage of an extension method on DynamoDbMapper that is automatically generated at build time. By using this approach, you don't need to refer to the schema by name. As shown in the following, the automatically generated getCarsTable extension method returns a reference to the table instance:

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

See the section called “Generate a schema” for more details and examples.

Invoke operations

DynamoDB Mapper supports a subset of the operations available on the SDK's DynamoDbClient. Mapper operations are named the same as their counterparts on the SDK client. Many mapper request/response members are the same as their SDK client counterparts, although some have been renamed, re-typed, or dropped altogether.

You invoke an operation on a table instance using a DSL syntax as shown in the following:

```
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)
```

You can also invoke an operation by using an explicit request object:

```
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)
```

The previous two code examples are equivalent.

Work with paginated responses

Some operations like `query` and `scan` can return data collections that might be too large to return in a single response. To ensure that all objects are processed, DynamoDB Mapper provides paginating methods, which do not call DynamoDB immediately, but instead return a `Flow` of the operation response type, such as `Flow<ScanResponse<Car>>` shown in the following:

```
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) }
}
```

Often, a flow of objects is more useful to business logic than a flow of responses *containing* objects. The mapper provides an extension method on paginated responses to access the flow of objects. For example, the following code returns a `Flow<Car>` rather than a `Flow<ScanResponse<Car>>` as shown previously:

```
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) }
```

Configure DynamoDB Mapper

> ⚠ **DynamoDB Mapper is a Developer Preview release. It is not feature complete and is subject to change.**

DynamoDB Mapper offers configuration options that you can use customize the behavior of the library to fit your application.

Use interceptors

The DynamoDB Mapper library defines hooks that you can tap into at critical stages of the mapper's request pipeline. You can implement the `Interceptor` interface to implement hooks to observe or modify the mapper process.

You can register one or more interceptors on a single DynamoDB Mapper as a configuration option. See the example at the end of this section for how you register an interceptor.

Understand the request pipeline

The mapper's request pipeline consist of the following 5 steps:

1. **Initialization:** Set up the operation and gathering initial context.

2. **Serialization:** Convert high-level request objects into low-level request objects. This step converts high-level Kotlin objects into DynamoDB items that consist of attribute names and values.
3. **Low-level invocation:** Execute a request on the underlying DynamoDB client.
4. **Deserialization:** Convert low-level response objects into high-level response objects. This step includes converting DynamoDB items that consist of attribute names and values into high-level Kotlin objects.
5. **Completion:** Finalize the high-level response to return to the caller. If an exception was thrown during the execution of the pipeline, this step finalizes the exception that is thrown to the caller.

Hooks

Hooks are interceptor methods that the mapper invokes before or after specific steps in the pipeline. There are two variants of hooks: *read-only* and *modify* (or read-write). For example, `readBeforeInvocation` is a read-only hook that the mapper executes in the phase before the low-level invocation step.

Read-only hooks

The mapper invokes read-only hooks before and after each step in the pipeline (except before the *Initialization* step and after the *Completion* step). Read-only hoods offer a read-only view of a high-level operation in progress. They provide a mechanism to examine the state of an operation for logging, debugging, collecting metrics, for example. Each read-only hook receives a context argument and returns `Unit`.

The mapper catches any exception that is thrown during a read-only hook and adds it to the context. It then passes the context with the exception to subsequent interceptor hooks in the same phase. The mapper throws any exception to the caller only after it calls the last interceptor's read-only hook for the same phase. For example, if a mapper is configured with two interceptors `A` and `B`, and `A`'s `readAfterSerialization` hook throws an exception, the mapper adds the exception to the context passed to `B`'s `readAfterSerialization` hook. After `B`'s `readAfterSerialization` hook has completed, the mapper throws the exception back to the caller.

Modify hooks

The mapper invokes modify hooks before each step in the pipeline (except before *Initialization*). Modify hooks offer the ability to see and modify a high-level operation in progress. They can be

used to customize behavior and data in ways that mapper configuration and item schemas do not. Each modify hook receives a context argument and returns some subset of that context as a result —either modified by the hook or passed-through from the input context.

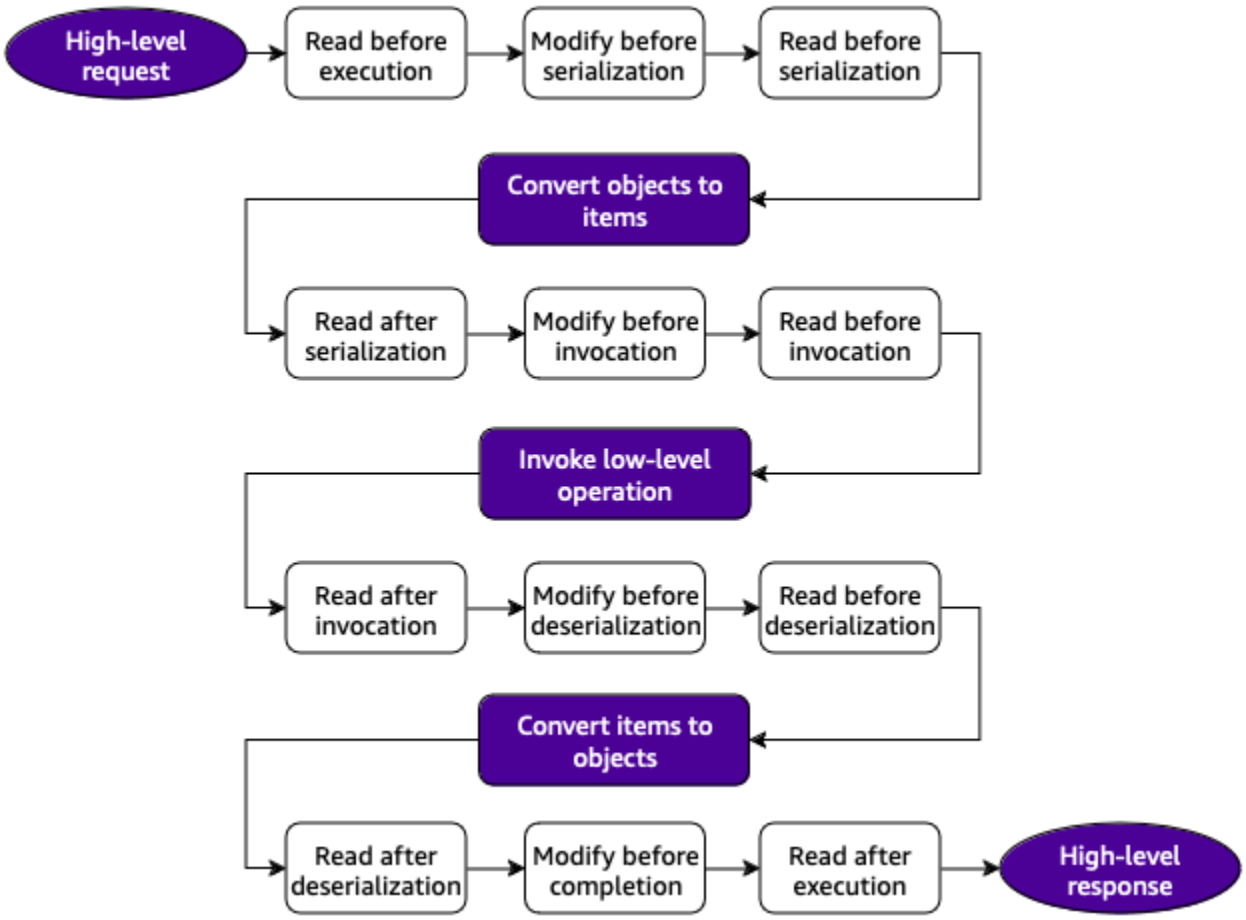
If the mapper catches any exception while it executes a modify hook, it doesn't execute any other interceptors' modify hooks in the same phase. The mapper adds the exception to the context and passes it to the next read-only hook. The mapper throws any exception to the caller only after it calls the last interceptors' read-only hook for the same phase. For example, if a mapper is configured with two interceptors `A` and `B`, and `A`'s `modifyBeforeSerialization` hook throws an exception, `B`'s `modifyBeforeSerialization` hook will not be invoked. Interceptors `A`'s and `B`'s `readAfterSerialization` hook will execute, after which the exception will be thrown back to the caller.

Execution order

The order in which interceptors are defined in a mapper's configuration determines the order that the mapper calls the hooks:

- For phases *before* the *Low-level invocation* step, it executes hooks in the *same order* that they were added in the configuration.
- For phases *after* the *Low-level invocation* step, it executes hooks in the *reverse order* from the order they were added in the configuration.

The following diagram shows the execution order of hook methods:



Text description of the execution order of hook methods

A mapper executes an interceptor's hooks in the following order:

1. DynamoDB Mapper invokes a high-level request
2. Read before execution
3. Modify before serialization
4. Read before serialization
5. DynamoDB Mapper converts objects to items
6. Read after serialization
7. Modify before invocation
8. Read before invocation
9. DynamoDB Mapper invokes the low-level operation
10. Read after invocation
11. Modify before deserialization

12Read before deserialization

13DynamoDB Mapper converts items to objects

14Read after deserialization

15Modify before completion

16Read after execution

17DynamoDB Mapper returns a high-level response

Example configuration

The following example shows how to configure an interceptor on a `DynamoDbMapper` instance:

```
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
}
```

Generate a schema from annotations

> ⚠ **DynamoDB Mapper is a Developer Preview release. It is not feature complete and is subject to change.**

DynamoDB Mapper relies on schemas that define the mapping between your Kotlin classes and DynamoDB items. Your Kotlin classes can drive the creation of schemas by using the schema generator Gradle plugin.

Apply the plugin

To start code generating schemas for your classes, apply the plugin in your application's build script and add a dependency on the annotations module. The following Gradle script snippet shows the necessary setup for code generation.

(You can navigate to the *X.Y.Z* link to see the latest version available.)

```
// 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")
}
```

Configure the plugin

The plugin offers a number of configuration options that you can apply by using the `dynamoDbMapper { ... }` plugin extension in your build script:

Option	Option description	Values
`generateBuilderClasses`	Controls whether DSL-style builder classes will be generated for classes annotated with `@DynamoDbItem`	`WHEN_REQUIRED` (default): Builder classes will not be generated for classes which consist of only public mutable members and have a zero-arg constructor `ALWAYS`: Builder classes will always be generated

Option	Option description	Values
`visibility`	Controls the visibility of generated classes	`PUBLIC` (default) `INTERNAL`
`destinationPackage`	Specifies the package name for generated classes	`RELATIVE` (default): Schema classes will be generated in a sub-package relative to your annotated class. By default, the sub-package is named `dynamodbmapper.generatedschemas`, and this is configurable by passing a string parameter `ABSOLUTE`: Schema classes will be generated in an absolute package relative to the root of your application. By default, the package is named `aws.sdk.kotlin.hll.dynamodbmapper.generatedschemas`, and this is configurable by passing a string parameter.
`generateGetTableExtension`	Controls whether a `DynamoDbMapper.get${CLASS_NAME}Table` extension method will be generated	`true` (default) `false`

Example Example of code-generation plugin configuration

This following example configures the destination package and visibility of the generated schema:

```
// 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
}
```

Annotate classes

The schema generator looks for class annotations to determine which classes to generate schemas for. To opt in to generating schemas, annotate your classes with @DynamoDbItem. You must also annotate a class property which serves as the item's partition key with the @DynamoDbPartitionKey annotation.

The following class definition shows the minimally required annotations for schema generation:

Example

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

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

Class annotations

The following annotations are applied to classes to control schema generation:

- `@DynamoDbItem`: Specifies that this class/interface describes an item type in a table. All public properties of this type will be mapped to attributes unless they are explicitly ignored. When present, a schema will be generated for this class.
  - `converterName`: An optional parameter which indicates a custom schema should be used rather than the one created by the schema generator plugin. This is the fully qualified name

of the custom `ItemConverter` class. The the section called "Define a custom item converter" section shows an example of creating and using a custom schema.

Property annotations

You can apply the following annotations to class properties to control schema generation:

- @`DynamoDbPartitionKey`: Specifies the partition key for the item.
- @`DynamoDbSortKey`: Specifies an optional sort key for the item.
- @`DynamoDbIgnore`: Specifies that this class property should not be converted to/from an Item attribute by the DynamoDB Mapper.
- @`DynamoDbAttribute`: Specifies an optional custom attribute name for this class property.

Define a custom item converter

In some cases, you may want to define a custom item converter for your class. One reason for this would be if your class uses a type that's not supported by the schema generator plugin. We use the following version of the `Employee` class as an example:

```
import kotlin.uuid.Uuid

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

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

The `Employee` class now uses a `kotlin.uuid.Uuid` type, which is not currently supported by the schema generator. Schema generation fails with an error: `Unsupported attribute type TypeRef(pkg=kotlin.uuid, shortName=Uuid, genericArgs=[], nullable=false)`. This error indicates that the plugin cannot generate an item converter for this class. Therefore, we need to write our own.

To do this, we implement an `ItemConverter` for the class, then modify the `@DynamoDbItem` class annotation by specifying the fully qualified name of the new item converter.

First, we implement a `ValueConverter` for the `kotlin.uuid.Uuid` class:

```
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())
}
```

Then, we implement an `ItemConverter` for our `Employee` class. The `ItemConverter` uses this new value converter in the attribute descriptor for "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
        )
    ),
)
```

Now that we have defined the item converter, we can apply it to our class. We update the @DynamoDbItem annotation to reference the item converter by providing the fully-qualified class name as shown in the following:

```
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
)
```

Finally we can begin using the class with DynamoDB Mapper.

Manually define schemas

DynamoDB Mapper is a Developer Preview release. It is not feature complete and is subject to change.

Define a schema in code

For maximum control and customizability, you can manually define and customize schemas in code.

As shown in the following snippet, you need to include fewer dependencies in your `build.gradle.kts` file compared to using annotation-driven schema creation.

(You can navigate to the *X.Y.Z* link to see the latest version available.)

```
// 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.
}
```

Note that you don't need the schema generator plugin nor the annotation package.

The mapping between a Kotlin class and a DynamoDB item requires an `ItemSchema<T>` implementation, where T is the type of the Kotlin class. A schema consists of the following elements:

- An item converter, which defines how to convert between Kotlin object instances and DynamoDB items.
- A partition key specification, which defines the name and type of the partition key attribute.
- Optionally, a sort key specification, which defines the name and type of the sort key attribute.

In the following code we manually create a `CarSchema` instance:

```
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)
```

The previous code creates a converter named `carConverter`, which is defined as an anonymous implementation of `ItemConverter<Car>`. The converter's `convertTo` method accepts a `Car` argument and returns an `Item` instance representing the literal keys and values of DynamoDB item attributes. The converter's `convertFrom` method accepts an `Item` argument and returns a `Car` instance from the attribute values of the `Item` argument.

Next the code creates two key specifications: one for the partition key and one for the sort key. Every DynamoDB table or index must have exactly one partition key and, correspondingly, so must every DynamoDB Mapper schema definition. Schemas may also have one sort key.

In the last statement, the code creates a schema for the `cars` DynamoDB table from the converter and key specifications.

The resulting schema is equivalent to the annotation-driven schema that we generated in the the section called "Define a schema with class annotations" section. For reference, the following is the annotated class we used:

Car class with DynamoDB Mapper annotations

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

```
    @DynamoDbSortKey
    val model: String,

    val initialYear: Int
)
```

In addition to implementing your own `ItemConverter`, DynamoDB Mapper includes several helpful implementations such as:

- `SimpleItemConverter`: provides simple conversion logic by using a builder class and attribute descriptors. See the example in the the section called "Define a custom item converter" for how you can make use of this implementation.
- `HeterogeneousItemConverter`: provides polymorphic type conversion logic by using a discriminator attribute and delegate `ItemConverter` instances for subtypes.
- `DocumentConverter`: provides conversion logic for unstructured data in `Document` objects.

Use secondary indices with DynamoDB Mapper

> ⚠ **DynamoDB Mapper is a Developer Preview release. It is not feature complete and is subject to change.**

Define a schema for a secondary index

DynamoDB tables support secondary indices which provide access to data using different keys from those defined on the base table itself. As with base tables, DynamoDB Mapper interacts with indices by using the `ItemSchema` type.

DynamoDB secondary indices are not required to contain every attribute from the base table. Accordingly, the Kotlin class that maps to an index may differ from the Kotlin class that maps to that index's base table. When that is the case, a separate schema must be declared for the index class.

The following code manually creates an index schema for the DynamoDB `cars` table.

```
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
        )
*/
```

We can now use `Model` instances in operations.

Use secondary indices in operations

DynamoDB Mapper supports a subset of operations on indices, namely `queryPaginated` and `scanPaginated`. To invoke these operations on an index, you must first obtain a reference to an index from the table object. In the following sample, we use the `modelSchema` that we created previously for the `cars-by-model` index (creation not shown here):

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

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

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

Use expressions

> **DynamoDB Mapper is a Developer Preview release. It is not feature complete and is subject to change.**

Certain DynamoDB operations accept expressions that you can use to specify constraints or conditions. DynamoDB Mapper provides an idiomatic Kotlin DSL to create expressions. The DSL brings greater structure and readability to your code and also makes it easier to write expressions.

This section describes the DSL syntax and provides various examples.

Use expressions in operations

You use expressions in operations like `scan`, where they filter the returned items based on criteria that you define. To use expressions with DynamoDB Mapper, add the expression component in the operation request.

The following snippet shows an example of a filter expression that is used in a `scan` operation. It uses a lambda argument to describe the filter criteria that limits the items to be returned to those with a `year` attribute value of 2001:

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

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

The following example shows a `query` operation that supports expressions in two places—sort key filtering and non-key filtering:

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

The previous code filters results to those that meet all three criteria:

- Partition key attribute value is 1000 *-AND-*
- Sort key attribute value starts with the letter *M* *-AND-*
- year attribute value is 2001

DSL components

The DSL syntax exposes several types of components—described below—that you use to build expressions.

Attributes

Most conditions reference attributes, which are identified by their key or document path. With the DSK, you create all attribute references by using the `attr` function and optionally make additional modifications.

The following code shows a range of example attribute references from simple to complex, such as list dereferencing by index and map dereferencing by key::

```
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`.
```

Equalities and inequalities

You can compare attribute values in an expression by equalities and inequalities. You can compare attribute values to literal values or other attribute values. The functions that you use to specify the conditions are:

- `eq`: is equal to (equivalent to ==)
- `neq`: is not equal to (equivalent to !=)
- `gt`: is greater than (equivalent to >)
- `gte`: is greater than or equal to (equivalent to >=)
- `lt`: is less than (equivalent to <)
- `lte`: is less than or equal to (equivalent to <=)

You combine the comparison function with arguments by using infix notation as shown in the following examples:

```
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`.
```

Ranges and sets

In addition to single values, you can compare attribute values to multiple values in ranges or sets. You use the infix `isIn` function to do the comparison as shown in the following examples:

```
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",
)
```

The `isIn` function provides overloads for collections (such as `Set<String>`) and for bounds that you can express as a Kotlin `ClosedRange<T>` (such as `IntRange`). For bounds that you cannot express as a `ClosedRange<T>` (such as byte arrays or other attribute references), you can use the `isBetween` function:

```
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`.
```

Boolean logic

You can combine individual conditions or altered using boolean logic by using the following functions:

- `and`: every condition must be true (equivalent to `&&`)
- `or`: at least one condition must be true (equivalent to `||`)
- `not`: the given condition must be false (equivalent to `!`)

The follow examples show each function:

```
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.
)
```

You can further combine boolean conditions by boolean functions to create nested logic as shown in the following expression:

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

The previous expression filters results to those that meet either of these conditions:

- Both of these conditions are true:
  - `foo` attribute value is 123 *-AND-*
  - `bar` attribute value is "abc"
- Both of these conditions are true:
  - `foo` attribute value is 234 *-AND-*
  - `bar` attribute value is "bcd"

This is equivalent to the following Kotlin boolean expression:

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

Functions and properties

The following functions and properties provide additional expression capabilities:

- `contains`: checks if a string/list attribute value contains a given value

- `exists`: checks if an attribute is defined and holds any value (including `null`)
- `notExists`: checks if an attribute is undefined
- `isOfType`: checks if an attribute value is of a given type, such as string, number, boolean, and so on
- `size`: gets the size of an attribute, such as the number of elements in a collection or the length of a string
- `startsWith`: checks if a string attribute value starts with a given substring

The following examples show use of additional functions and properties that you can use in expressions:

```
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.
```

Sort key filters

Filter expressions on sort keys (such as in the `query` operation's `keyCondition` parameter) do not use named attribute values. To use a sort key in a filter, you must use the keyword `sortKey` in all comparisons. The `sortKey` keyword replaces `attr("<sort key name>")` as shown in the following examples:

```
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).
```

You cannot combine sort key filters with boolean logic and they support only a subset of the comparisons described above:

- Equalities and inequalities: all comparisons supported
- Ranges and sets: all comparisons supported
- Boolean logic: not supported
- Functions and properties: only `startsWith` is supported

SDK for Kotlin code examples

The code examples in this topic show you how to use the AWS SDK for Kotlin with AWS.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Some services contain additional example categories that show how to leverage libraries or functions specific to the service.

Services

- API Gateway examples using SDK for Kotlin
- Aurora examples using SDK for Kotlin
- Auto Scaling examples using SDK for Kotlin
- Amazon Bedrock examples using SDK for Kotlin
- Amazon Bedrock Runtime examples using SDK for Kotlin
- CloudWatch examples using SDK for Kotlin
- CloudWatch Logs examples using SDK for Kotlin
- Amazon Cognito Identity Provider examples using SDK for Kotlin
- Amazon Comprehend examples using SDK for Kotlin
- DynamoDB examples using SDK for Kotlin
- Amazon EC2 examples using SDK for Kotlin
- Amazon ECR examples using SDK for Kotlin
- OpenSearch Service examples using SDK for Kotlin
- EventBridge examples using SDK for Kotlin
- AWS Glue examples using SDK for Kotlin
- IAM examples using SDK for Kotlin
- AWS IoT examples using SDK for Kotlin
- AWS IoT data examples using SDK for Kotlin

- AWS IoT FleetWise examples using SDK for Kotlin
- Amazon Keyspaces examples using SDK for Kotlin
- AWS KMS examples using SDK for Kotlin
- Lambda examples using SDK for Kotlin
- Amazon Location examples using SDK for Kotlin
- MediaConvert examples using SDK for Kotlin
- Amazon Pinpoint examples using SDK for Kotlin
- Amazon RDS examples using SDK for Kotlin
- Amazon RDS Data Service examples using SDK for Kotlin
- Amazon Redshift examples using SDK for Kotlin
- Amazon Rekognition examples using SDK for Kotlin
- Route 53 domain registration examples using SDK for Kotlin
- Amazon S3 examples using SDK for Kotlin
- SageMaker AI examples using SDK for Kotlin
- Secrets Manager examples using SDK for Kotlin
- Amazon SES examples using SDK for Kotlin
- Amazon SNS examples using SDK for Kotlin
- Amazon SQS examples using SDK for Kotlin
- Step Functions examples using SDK for Kotlin
- Support examples using SDK for Kotlin
- Amazon Translate examples using SDK for Kotlin

API Gateway examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with API Gateway.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Scenarios

Scenarios

Create a serverless application to manage photos

The following code example shows how to create a serverless application that lets users manage photos using labels.

SDK for Kotlin

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

Services used in this example

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

Aurora examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Aurora.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Basics
- Actions
- Scenarios

Basics

Learn the basics

The following code example shows how to:

- Create a custom Aurora DB cluster parameter group and set parameter values.
- Create a DB cluster that uses the parameter group.
- Create a DB instance that contains a database.
- Take a snapshot of the DB cluster, then clean up resources.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
Before running this 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Actions

`CreateDBCluster`

The following code example shows how to use `CreateDBCluster`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- For API details, see CreateDBCluster in *AWS SDK for Kotlin API reference*.

`CreateDBClusterParameterGroup`

The following code example shows how to use `CreateDBClusterParameterGroup`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- For API details, see CreateDBClusterParameterGroup in *AWS SDK for Kotlin API reference*.

`CreateDBClusterSnapshot`

The following code example shows how to use `CreateDBClusterSnapshot`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- For API details, see CreateDBClusterSnapshot in *AWS SDK for Kotlin API reference*.

`CreateDBInstance`

The following code example shows how to use `CreateDBInstance`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- For API details, see CreateDBInstance in *AWS SDK for Kotlin API reference*.

`DeleteDBCluster`

The following code example shows how to use `DeleteDBCluster`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- For API details, see DeleteDBCluster in *AWS SDK for Kotlin API reference*.

`DeleteDBClusterParameterGroup`

The following code example shows how to use `DeleteDBClusterParameterGroup`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { 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.")
    }
}
```

- For API details, see DeleteDBClusterParameterGroup in *AWS SDK for Kotlin API reference*.

`DeleteDBInstance`

The following code example shows how to use `DeleteDBInstance`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
```

```
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- For API details, see DeleteDBInstance in *AWS SDK for Kotlin API reference*.

`DescribeDBClusterParameterGroups`

The following code example shows how to use `DescribeDBClusterParameterGroups`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { 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}")
        }
```

```
    }
}
```

- For API details, see DescribeDBClusterParameterGroups in *AWS SDK for Kotlin API reference*.

`DescribeDBClusterParameters`

The following code example shows how to use `DescribeDBClusterParameters`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- For API details, see DescribeDBClusterParameters in *AWS SDK for Kotlin API reference*.

`DescribeDBClusterSnapshots`

The following code example shows how to use `DescribeDBClusterSnapshots`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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!")
}
```

- For API details, see DescribeDBClusterSnapshots in *AWS SDK for Kotlin API reference*.

`DescribeDBClusters`

The following code example shows how to use `DescribeDBClusters`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
```

```
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- For API details, see DescribeDBClusters in *AWS SDK for Kotlin API reference*.

`DescribeDBEngineVersions`

The following code example shows how to use `DescribeDBEngineVersions`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DescribeDBEngineVersions in *AWS SDK for Kotlin API reference*.

`DescribeDBInstances`

The following code example shows how to use `DescribeDBInstances`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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")
}
```

- For API details, see DescribeDBInstances in *AWS SDK for Kotlin API reference*.

`ModifyDBClusterParameterGroup`

The following code example shows how to use `ModifyDBClusterParameterGroup`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Modify 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.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- For API details, see ModifyDBClusterParameterGroup in *AWS SDK for Kotlin API reference*.

Scenarios

Create an Aurora Serverless work item tracker

The following code example shows how to create a web application that tracks work items in an Amazon Aurora Serverless database and uses Amazon Simple Email Service (Amazon SES) to send reports.

SDK for Kotlin

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on GitHub.

Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Auto Scaling examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Auto Scaling.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- Create an Amazon EC2 Auto Scaling group with a launch template and Availability Zones, and get information about running instances.
- Enable Amazon CloudWatch metrics collection.

- Update the group's desired capacity and wait for an instance to start.
- Terminate an instance in the group.
- List scaling activities that occur in response to user requests and capacity changes.
- Get statistics for CloudWatch metrics, then clean up resources.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Actions

`CreateAutoScalingGroup`

The following code example shows how to use `CreateAutoScalingGroup`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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!")
    }
}
```

- For API details, see CreateAutoScalingGroup in *AWS SDK for Kotlin API reference*.

`DeleteAutoScalingGroup`

The following code example shows how to use `DeleteAutoScalingGroup`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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")
    }
}
```

- For API details, see DeleteAutoScalingGroup in *AWS SDK for Kotlin API reference*.

`DescribeAutoScalingGroups`

The following code example shows how to use `DescribeAutoScalingGroups`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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)
            }
        }
    }
}
```

- For API details, see DescribeAutoScalingGroups in *AWS SDK for Kotlin API reference*.

`DescribeAutoScalingInstances`

The following code example shows how to use `DescribeAutoScalingInstances`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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}")
        }
    }
}
```

- For API details, see DescribeAutoScalingInstances in *AWS SDK for Kotlin API reference*.

`DescribeScalingActivities`

The following code example shows how to use `DescribeScalingActivities`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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}")
        }
    }
}
```

- For API details, see DescribeScalingActivities in *AWS SDK for Kotlin API reference*.

`DisableMetricsCollection`

The following code example shows how to use `DisableMetricsCollection`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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")
    }
```

```
}
```

- For API details, see DisableMetricsCollection in *AWS SDK for Kotlin API reference*.

`EnableMetricsCollection`

The following code example shows how to use `EnableMetricsCollection`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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")
    }
}
```

- For API details, see EnableMetricsCollection in *AWS SDK for Kotlin API reference*.

`SetDesiredCapacity`

The following code example shows how to use `SetDesiredCapacity`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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")
    }
}
```

- For API details, see SetDesiredCapacity in *AWS SDK for Kotlin API reference*.

`TerminateInstanceInAutoScalingGroup`

The following code example shows how to use `TerminateInstanceInAutoScalingGroup`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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")
    }
}
```

- For API details, see TerminateInstanceInAutoScalingGroup in *AWS SDK for Kotlin API reference*.

`UpdateAutoScalingGroup`

The following code example shows how to use `UpdateAutoScalingGroup`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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")
    }
}
```

- For API details, see UpdateAutoScalingGroup in *AWS SDK for Kotlin API reference*.

Amazon Bedrock examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Bedrock.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Actions

Actions

`ListFoundationModels`

The following code example shows how to use `ListFoundationModels`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

List the available Amazon Bedrock foundation models.

```
suspend fun listFoundationModels(): List<FoundationModelSummary>? {
    BedrockClient.fromEnvironment { 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
    }
}
```

- For API details, see ListFoundationModels in *AWS SDK for Kotlin API reference*.

Amazon Bedrock Runtime examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Bedrock Runtime.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Amazon Nova

Amazon Nova

Converse

The following code example shows how to send a text message to Amazon Nova, using Bedrock's Converse API.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Send a text message to Amazon Nova, using Bedrock's Converse API.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models to
 generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
suspend fun main() {
    converse().also { println(it) }
}
```

```
suspend fun converse(): String {
    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in one line."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
        }

        // Send the request and process the model's response
        runCatching {
            val response = client.converse(request)
            return response.output!!.asMessage().content.first().asText()
        }.getOrElse { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId",
 error)
        }
    }
}
```

- For API details, see Converse in *AWS SDK for Kotlin API reference*.

ConverseStream

The following code example shows how to send a text message to Amazon Nova, using Bedrock's Converse API and process the response stream in real-time.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Send a text message to Amazon Nova using Bedrock's Converse API and process the response stream in real-time.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamOutput
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message with a prompt
 * - Configure a streaming request with parameters
 * - Process the response stream in real time
 */
suspend fun main() {
    converseStream()
}

suspend fun converseStream(): String {
    // A buffer to collect the complete response
    val completeResponseBuffer = StringBuilder()

    // Create and configure the Bedrock runtime client
```

```
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in a
 paragraph."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseStreamRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
        }

        // Process the streaming response
        runCatching {
            client.converseStream(request) { response ->
                response.stream?.collect { chunk ->
                    when (chunk) {
                        is ConverseStreamOutput.ContentBlockDelta -> {
                            // Process each text chunk as it arrives
                            chunk.value.delta?.asText()?.let { text ->
                                print(text)
                                System.out.flush() // Ensure immediate output
                                completeResponseBuffer.append(text)
                            }
                        }
                        else -> {} // Other output block types can be handled as
 needed
                    }
                }
            }
```

```
        }.onFailure { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId:
 $error", error)
        }
    }

    return completeResponseBuffer.toString()
}
```

- For API details, see ConverseStream in *AWS SDK for Kotlin API reference*.

CloudWatch examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with CloudWatch.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello CloudWatch

The following code examples show how to get started using CloudWatch.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
Before running this 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.fromEnvironment { 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}")
            }
    }
}
```

- For API details, see ListMetrics in *AWS SDK for Kotlin API reference*.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- List CloudWatch namespaces and metrics.
- Get statistics for a metric and for estimated billing.
- Create and update a dashboard.
- Create and add data to a metric.
- Create and trigger an alarm, then view alarm history.
- Add an anomaly detector.
- Get a metric image, then clean up resources.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run an interactive scenario demonstrating CloudWatch features.

```
/**
 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

Actions

`DeleteAlarms`

The following code example shows how to use `DeleteAlarms`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- For API details, see DeleteAlarms in *AWS SDK for Kotlin API reference*.

`DeleteAnomalyDetector`

The following code example shows how to use `DeleteAnomalyDetector`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- For API details, see DeleteAnomalyDetector in *AWS SDK for Kotlin API reference*.

`DeleteDashboards`

The following code example shows how to use `DeleteDashboards`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteDashboard(dashboardName: String) {
```

```
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- For API details, see DeleteDashboards in *AWS SDK for Kotlin API reference*.

`DescribeAlarmHistory`

The following code example shows how to use `DescribeAlarmHistory`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- For API details, see DescribeAlarmHistory in *AWS SDK for Kotlin API reference*.

`DescribeAlarms`

The following code example shows how to use `DescribeAlarms`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
```

```
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DescribeAlarms in *AWS SDK for Kotlin API reference*.

`DescribeAlarmsForMetric`

The following code example shows how to use `DescribeAlarmsForMetric`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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.")
        }
    }
}
```

- For API details, see DescribeAlarmsForMetric in *AWS SDK for Kotlin API reference*.

`DescribeAnomalyDetectors`

The following code example shows how to use `DescribeAnomalyDetectors`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DescribeAnomalyDetectors in *AWS SDK for Kotlin API reference*.

`DisableAlarmActions`

The following code example shows how to use `DisableAlarmActions`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun disableActions(alarmName: String) {
    val request =
        DisableAlarmActionsRequest {
            alarmNames = listOf(alarmName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- For API details, see DisableAlarmActions in *AWS SDK for Kotlin API reference*.

`EnableAlarmActions`

The following code example shows how to use `EnableAlarmActions`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun enableActions(alarm: String) {
    val request =
        EnableAlarmActionsRequest {
            alarmNames = listOf(alarm)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- For API details, see EnableAlarmActions in *AWS SDK for Kotlin API reference*.

`GetMetricData`

The following code example shows how to use `GetMetricData`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see GetMetricData in *AWS SDK for Kotlin API reference*.

`GetMetricStatistics`

The following code example shows how to use `GetMetricStatistics`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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")
            }
        }
    }
}
```

- For API details, see GetMetricStatistics in *AWS SDK for Kotlin API reference*.

`GetMetricWidgetImage`

The following code example shows how to use `GetMetricWidgetImage`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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")
}
```

- For API details, see GetMetricWidgetImage in *AWS SDK for Kotlin API reference*.

`ListDashboards`

The following code example shows how to use `ListDashboards`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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}")
            }
    }
}
```

- For API details, see ListDashboards in *AWS SDK for Kotlin API reference*.

`ListMetrics`

The following code example shows how to use `ListMetrics`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { 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
}
```

- For API details, see ListMetrics in *AWS SDK for Kotlin API reference*.

`PutAnomalyDetector`

The following code example shows how to use `PutAnomalyDetector`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}
```

- For API details, see PutAnomalyDetector in *AWS SDK for Kotlin API reference*.

`PutDashboard`

The following code example shows how to use `PutDashboard`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }
```

```
    CloudWatchClient.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- For API details, see PutDashboard in *AWS SDK for Kotlin API reference*.

`PutMetricAlarm`

The following code example shows how to use `PutMetricAlarm`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- For API details, see PutMetricAlarm in *AWS SDK for Kotlin API reference*.

`PutMetricData`

The following code example shows how to use `PutMetricData`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- For API details, see PutMetricData in *AWS SDK for Kotlin API reference*.

CloudWatch Logs examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with CloudWatch Logs.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Actions

Actions

`DeleteSubscriptionFilter`

The following code example shows how to use `DeleteSubscriptionFilter`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { logs ->
```

```
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- For API details, see DeleteSubscriptionFilter in *AWS SDK for Kotlin API reference*.

`DescribeSubscriptionFilters`

The following code example shows how to use `DescribeSubscriptionFilters`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DescribeSubscriptionFilters in *AWS SDK for Kotlin API reference*.

`StartLiveTail`

The following code example shows how to use `StartLiveTail`.

SDK for Kotlin

Include the required files.

```
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
```

Start the Live Tail session.

```
    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)
    }
```

- For API details, see StartLiveTail in *AWS SDK for Kotlin API reference*.

Amazon Cognito Identity Provider examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Cognito Identity Provider.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Actions
- Scenarios

Actions

`AdminGetUser`

The following code example shows how to use `AdminGetUser`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- For API details, see AdminGetUser in *AWS SDK for Kotlin API reference*.

`AdminInitiateAuth`

The following code example shows how to use `AdminInitiateAuth`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- For API details, see AdminInitiateAuth in *AWS SDK for Kotlin API reference*.

`AdminRespondToAuthChallenge`

The following code example shows how to use `AdminRespondToAuthChallenge`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Respond to an authentication challenge.
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.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- For API details, see AdminRespondToAuthChallenge in *AWS SDK for Kotlin API reference*.

`AssociateSoftwareToken`

The following code example shows how to use `AssociateSoftwareToken`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { 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
    }
}
```

- For API details, see AssociateSoftwareToken in *AWS SDK for Kotlin API reference*.

`ConfirmSignUp`

The following code example shows how to use `ConfirmSignUp`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- For API details, see ConfirmSignUp in *AWS SDK for Kotlin API reference*.

`ListUsers`

The following code example shows how to use `ListUsers`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }
```

```
    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- For API details, see ListUsers in *AWS SDK for Kotlin API reference*.

`ResendConfirmationCode`

The following code example shows how to use `ResendConfirmationCode`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- For API details, see ResendConfirmationCode in *AWS SDK for Kotlin API reference*.

`SignUp`

The following code example shows how to use `SignUp`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
```

```
        println("User has been signed up")
    }
}
```

- For API details, see SignUp in *AWS SDK for Kotlin API reference*.

`VerifySoftwareToken`

The following code example shows how to use `VerifySoftwareToken`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- For API details, see VerifySoftwareToken in *AWS SDK for Kotlin API reference*.

Scenarios

Sign up a user with a user pool that requires MFA

The following code example shows how to:

- Sign up and confirm a user with a username, password, and email address.
- Set up multi-factor authentication by associating an MFA application with the user.
- Sign in by using a password and an MFA code.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 Before running this 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp

- VerifySoftwareToken

Amazon Comprehend examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Comprehend.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Scenarios

Scenarios

Create a messaging application

The following code example shows how to create a messaging application by using Amazon SQS.

SDK for Kotlin

Shows how to use the Amazon SQS API to develop a Spring REST API that sends and retrieves messages.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Comprehend
- Amazon SQS

DynamoDB examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with DynamoDB.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Basics
- Actions
- Scenarios

Basics

Learn the basics

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table, then delete the table.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a DynamoDB table.

```
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 request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { 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}")
    }
}
```

Create a helper function to download and extract the sample JSON file.

```
// 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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}
```

Get an item from a 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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

Full example.

```
suspend fun main() {
    val tableName = "Movies"
    val fileName = "../../../resources/sample_files/movies.json"
    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 request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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]}")
            }
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Actions

`CreateTable`

The following code example shows how to use `CreateTable`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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 request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef)
            keySchema = listOf(keySchemaVal)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { 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
```

```
    }
}
```

- For API details, see CreateTable in *AWS SDK for Kotlin API reference*.

`DeleteItem`

The following code example shows how to use `DeleteItem`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- For API details, see DeleteItem in *AWS SDK for Kotlin API reference*.

`DeleteTable`

The following code example shows how to use `DeleteTable`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- For API details, see DeleteTable in *AWS SDK for Kotlin API reference*.

`GetItem`

The following code example shows how to use `GetItem`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

- For API details, see GetItem in *AWS SDK for Kotlin API reference*.

`ListTables`

The following code example shows how to use `ListTables`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listAllTables() {
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
```

```
            println("Table name is $tableName")
        }
    }
}
```

- For API details, see ListTables in *AWS SDK for Kotlin API reference*.

`PutItem`

The following code example shows how to use `PutItem`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- For API details, see PutItem in *AWS SDK for Kotlin API reference*.

`Query`

The following code example shows how to use `Query`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

- For API details, see Query in *AWS SDK for Kotlin API reference*.

Scan

The following code example shows how to use `Scan`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun scanItems(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { 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]}")
            }
        }
    }
}
```

- For API details, see Scan in *AWS SDK for Kotlin API reference*.

`UpdateItem`

The following code example shows how to use `UpdateItem`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
```

```
}
```

- For API details, see UpdateItem in *AWS SDK for Kotlin API reference*.

Scenarios

Create a serverless application to manage photos

The following code example shows how to create a serverless application that lets users manage photos using labels.

SDK for Kotlin

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

Services used in this example

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

Create a web application to track DynamoDB data

The following code example shows how to create a web application that tracks work items in an Amazon DynamoDB table and uses Amazon Simple Email Service (Amazon SES) to send reports.

SDK for Kotlin

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- DynamoDB
- Amazon SES

Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { 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 request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            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")
    }
}
```

- For API details, see BatchExecuteStatement in *AWS SDK for Kotlin API reference*.

Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.

- Delete an item by running a DELETE statement.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { region = "us-east-1" }
    val tableName = "MoviesPartiQ"
    val fileName = "../../../resources/sample_files/movies.json"
    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 request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            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)
}
```

- For API details, see ExecuteStatement in *AWS SDK for Kotlin API reference*.

Amazon EC2 examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon EC2.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon EC2

The following code examples show how to get started using Amazon EC2.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
```

```
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DescribeSecurityGroups in *AWS SDK for Kotlin API reference*.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- Create a key pair and security group.
- Select an Amazon Machine Image (AMI) and compatible instance type, then create an instance.
- Stop and restart the instance.
- Associate an Elastic IP address with your instance.
- Connect to your instance with SSH, then clean up resources.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 Before running this 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress

- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Actions

`AllocateAddress`

The following code example shows how to use `AllocateAddress`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client.fromEnvironment { 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
```

```
    }
}
```

- For API details, see AllocateAddress in *AWS SDK for Kotlin API reference*.

`AssociateAddress`

The following code example shows how to use `AssociateAddress`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}
```

- For API details, see AssociateAddress in *AWS SDK for Kotlin API reference*.

`AuthorizeSecurityGroupIngress`

The following code example shows how to use `AuthorizeSecurityGroupIngress`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { 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
    }
}
```

- For API details, see AuthorizeSecurityGroupIngress in *AWS SDK for Kotlin API reference*.

`CreateKeyPair`

The following code example shows how to use `CreateKeyPair`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- For API details, see CreateKeyPair in *AWS SDK for Kotlin API reference*.

`CreateSecurityGroup`

The following code example shows how to use `CreateSecurityGroup`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { 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
    }
}
```

- For API details, see CreateSecurityGroup in *AWS SDK for Kotlin API reference*.

`DeleteKeyPair`

The following code example shows how to use `DeleteKeyPair`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
```

```
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- For API details, see DeleteKeyPair in *AWS SDK for Kotlin API reference*.

`DeleteSecurityGroup`

The following code example shows how to use `DeleteSecurityGroup`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- For API details, see DeleteSecurityGroup in *AWS SDK for Kotlin API reference*.

`DescribeInstanceTypes`

The following code example shows how to use `DescribeInstanceTypes`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Get a list of instance types.
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.fromEnvironment { 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
    }
}
```

- For API details, see DescribeInstanceTypes in *AWS SDK for Kotlin API reference*.

`DescribeInstances`

The following code example shows how to use `DescribeInstances`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client.fromEnvironment { 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}")
            }
        }
    }
}
```

- For API details, see DescribeInstances in *AWS SDK for Kotlin API reference*.

`DescribeKeyPairs`

The following code example shows how to use `DescribeKeyPairs`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeEC2Keys() {
    Ec2Client.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DescribeKeyPairs in *AWS SDK for Kotlin API reference*.

`DescribeSecurityGroups`

The following code example shows how to use `DescribeSecurityGroups`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DescribeSecurityGroups in *AWS SDK for Kotlin API reference*.

`DisassociateAddress`

The following code example shows how to use `DisassociateAddress`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- For API details, see DisassociateAddress in *AWS SDK for Kotlin API reference*.

`ReleaseAddress`

The following code example shows how to use `ReleaseAddress`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- For API details, see ReleaseAddress in *AWS SDK for Kotlin API reference*.

`RunInstances`

The following code example shows how to use `RunInstances`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createEC2Instance(
    name: String,
    amiId: String,
): String? {
```

```
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client.fromEnvironment { 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
    }
}
```

- For API details, see RunInstances in *AWS SDK for Kotlin API reference*.

`StartInstances`

The following code example shows how to use `StartInstances`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { 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")
    }
}
```

- For API details, see StartInstances in *AWS SDK for Kotlin API reference*.

`StopInstances`

The following code example shows how to use `StopInstances`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { 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")
    }
}
```

- For API details, see StopInstances in *AWS SDK for Kotlin API reference*.

`TerminateInstances`

The following code example shows how to use `TerminateInstances`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see TerminateInstances in *AWS SDK for Kotlin API reference*.

Amazon ECR examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon ECR.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon ECR

The following code examples show how to get started using Amazon ECR.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import 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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
        }
    }
}
```

- For API details, see listImages in *AWS SDK for Kotlin API reference*.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- Create an Amazon ECR repository.
- Set repository policies.
- Retrieve repository URIs.
- Get Amazon ECR authorization tokens.
- Set lifecycle policies for Amazon ECR repositories.

- Push a Docker image to an Amazon ECR repository.
- Verify the existence of an image in an Amazon ECR repository.
- List Amazon ECR repositories for your account and get details about them.
- Delete Amazon ECR repositories.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run an interactive scenario demonstrating Amazon ECR features.

```
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:
 *
 * /scenarios/basics/ecr/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 = args[0]
    var localImageName: String
    var accountId = args[1]
    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.")
```

```
        }
    }
}
```

A wrapper class for Amazon ECR SDK methods.

```
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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)
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateRepository

- DeleteRepository
- DescribeImages
- DescribeRepositories
- GetAuthorizationToken
- GetRepositoryPolicy
- SetRepositoryPolicy
- StartLifecyclePolicyPreview

Actions

`CreateRepository`

The following code example shows how to use `CreateRepository`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /**
    * 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.fromEnvironment { 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
        }
    }
```

- For API details, see CreateRepository in *AWS SDK for Kotlin API reference*.

`DeleteRepository`

The following code example shows how to use `DeleteRepository`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples 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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- For API details, see DeleteRepository in *AWS SDK for Kotlin API reference*.

`DescribeImages`

The following code example shows how to use `DescribeImages`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * 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.fromEnvironment { 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.")
            }
        }
    }
```

- For API details, see DescribeImages in *AWS SDK for Kotlin API reference*.

`DescribeRepositories`

The following code example shows how to use `DescribeRepositories`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * 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.fromEnvironment { 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 ""
            }
        }
    }
```

- For API details, see DescribeRepositories in *AWS SDK for Kotlin API reference*.

`GetAuthorizationToken`

The following code example shows how to use `GetAuthorizationToken`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { 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.")
            }
        }
    }
```

- For API details, see GetAuthorizationToken in *AWS SDK for Kotlin API reference*.

`GetRepositoryPolicy`

The following code example shows how to use `GetRepositoryPolicy`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * Gets 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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- For API details, see GetRepositoryPolicy in *AWS SDK for Kotlin API reference*.

`PushImageCmd`

The following code example shows how to use `PushImageCmd`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * 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.fromEnvironment { 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)
            }
        }
    }
```

- For API details, see PushImageCmd in *AWS SDK for Kotlin API reference*.

`SetRepositoryPolicy`

The following code example shows how to use `SetRepositoryPolicy`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * 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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
```

```
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- For API details, see SetRepositoryPolicy in *AWS SDK for Kotlin API reference*.

`StartLifecyclePolicyPreview`

The following code example shows how to use `StartLifecyclePolicyPreview`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * 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.fromEnvironment { 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.")
            }
        }
    }
```

- For API details, see StartLifecyclePolicyPreview in *AWS SDK for Kotlin API reference*.

OpenSearch Service examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with OpenSearch Service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Actions

Actions

`CreateDomain`

The following code example shows how to use `CreateDomain`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val createResponse = searchClient.createDomain(request)
        println("Domain status is ${createResponse.domainStatus}")
        println("Domain Id is ${createResponse.domainStatus?.domainId}")
    }
}
```

- For API details, see CreateDomain in *AWS SDK for Kotlin API reference*.

`DeleteDomain`

The following code example shows how to use `DeleteDomain`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteSpecificDomain(domainNameVal: String) {
    val request =
        DeleteDomainRequest {
            domainName = domainNameVal
        }
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        searchClient.deleteDomain(request)
        println("$domainNameVal was successfully deleted.")
    }
}
```

- For API details, see DeleteDomain in *AWS SDK for Kotlin API reference*.

`ListDomainNames`

The following code example shows how to use `ListDomainNames`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listAllDomains() {
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val response: ListDomainNamesResponse =
 searchClient.listDomainNames(ListDomainNamesRequest {})
        response.domainNames?.forEach { domain ->
            println("Domain name is " + domain.domainName)
        }
    }
}
```

- For API details, see ListDomainNames in *AWS SDK for Kotlin API reference*.

`UpdateDomainConfig`

The following code example shows how to use `UpdateDomainConfig`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            instanceCount = 3
        }

    val request =
```

```
        UpdateDomainConfigRequest {
            domainName = domainNameVal
            clusterConfig = clusterConfigOb
        }

    println("Sending domain update request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val updateResponse = searchClient.updateDomainConfig(request)
        println("Domain update response from Amazon OpenSearch Service:")
        println(updateResponse.toString())
    }
}
```

- For API details, see UpdateDomainConfig in *AWS SDK for Kotlin API reference*.

EventBridge examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with EventBridge.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello EventBridge

The following code examples show how to get started using EventBridge.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import 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.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see ListEventBuses in *AWS SDK for Kotlin API reference*.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- Create a rule and add a target to it.
- Enable and disable rules.
- List and update rules and targets.

- Send events, then clean up resources.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/*
 Before running this 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
    }
}
```

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }
```

```
    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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"
```

```
            }
        ]
    }
    """.trimIndent()

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest =
        CreateTopicRequest {
            name = topicName
            attributes = topicAttributes
        }

    SnsClient.fromEnvironment { 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.fromEnvironment { 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"]
            }
        }
    }
    """.trimIndent()

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Actions

`DeleteRule`

The following code example shows how to use `DeleteRule`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}
```

- For API details, see DeleteRule in *AWS SDK for Kotlin API reference*.

`DescribeRule`

The following code example shows how to use `DescribeRule`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

- For API details, see DescribeRule in *AWS SDK for Kotlin API reference*.

`DisableRule`

The following code example shows how to use `DisableRule`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- For API details, see DisableRule in *AWS SDK for Kotlin API reference*.

`EnableRule`

The following code example shows how to use `EnableRule`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- For API details, see EnableRule in *AWS SDK for Kotlin API reference*.

`ListRuleNamesByTarget`

The following code example shows how to use `ListRuleNamesByTarget`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
```

```
}
```

- For API details, see ListRuleNamesByTarget in *AWS SDK for Kotlin API reference*.

`ListRules`

The following code example shows how to use `ListRules`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see ListRules in *AWS SDK for Kotlin API reference*.

`ListTargetsByRule`

The following code example shows how to use `ListTargetsByRule`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

- For API details, see ListTargetsByRule in *AWS SDK for Kotlin API reference*.

`PutEvents`

The following code example shows how to use `PutEvents`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putEvents(eventsRequest)
    }
}
```

- For API details, see PutEvents in *AWS SDK for Kotlin API reference*.

`PutRule`

The following code example shows how to use `PutRule`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a scheduled rule.

```
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.fromEnvironment { region = "us-west-2" }.use { eventBrClient -
>
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

Create a rule that triggers when an object is added to an Amazon Simple Storage Service bucket.

```
// 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"]
            }
        }
    }
    """.trimIndent()
```

```
    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- For API details, see PutRule in *AWS SDK for Kotlin API reference*.

`PutTargets`

The following code example shows how to use `PutTargets`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// 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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
}
```

Add an input transformer to a target for a rule.

```
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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- For API details, see PutTargets in *AWS SDK for Kotlin API reference*.

`RemoveTargets`

The following code example shows how to use `RemoveTargets`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient.fromEnvironment { 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")
            }
        }
    }
}
```

- For API details, see RemoveTargets in *AWS SDK for Kotlin API reference*.

AWS Glue examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS Glue.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- Create a crawler that crawls a public Amazon S3 bucket and generates a database of CSV-formatted metadata.

- List information about databases and tables in your AWS Glue Data Catalog.
- Create a job to extract CSV data from the S3 bucket, transform the data, and load JSON-formatted output into another S3 bucket.
- List information about job runs, view transformed data, and clean up resources.

For more information, see Tutorial: Getting started with AWS Glue Studio.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { glueClient ->
```

```
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
    val request =
        GetJobsRequest {
            maxResults = 10
        }

    GlueClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns

- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Actions

`CreateCrawler`

The following code example shows how to use `CreateCrawler`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- For API details, see CreateCrawler in *AWS SDK for Kotlin API reference*.

`GetCrawler`

The following code example shows how to use `GetCrawler`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getSpecificCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
    GlueClient.fromEnvironment { 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")
    }
```

```
}
```

- For API details, see GetCrawler in *AWS SDK for Kotlin API reference*.

`GetDatabase`

The following code example shows how to use `GetDatabase`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- For API details, see GetDatabase in *AWS SDK for Kotlin API reference*.

`StartCrawler`

The following code example shows how to use `StartCrawler`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- For API details, see StartCrawler in *AWS SDK for Kotlin API reference*.

IAM examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with IAM.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to create a user and assume a role.

> **Warning**
>
> To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as AWS IAM Identity Center.

- Create a user with no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list S3 buckets using temporary credentials, then clean up resources.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create functions that wrap IAM user actions.

```
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 = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:*"
                ],
                "Resource": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment {
        region = "us-east-1"
        credentialsProvider = staticCredentials
    }

    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.fromEnvironment { 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.fromEnvironment { 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)
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Actions

`AttachRolePolicy`

The following code example shows how to use `AttachRolePolicy`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { 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
}
```

- For API details, see AttachRolePolicy in *AWS SDK for Kotlin API reference*.

`CreateAccessKey`

The following code example shows how to use `CreateAccessKey`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
```

```
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- For API details, see CreateAccessKey in *AWS SDK for Kotlin API reference*.

`CreateAccountAlias`

The following code example shows how to use `CreateAccountAlias`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- For API details, see CreateAccountAlias in *AWS SDK for Kotlin API reference*.

`CreatePolicy`

The following code example shows how to use `CreatePolicy`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- For API details, see CreatePolicy in *AWS SDK for Kotlin API reference*.

`CreateUser`

The following code example shows how to use `CreateUser`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- For API details, see CreateUser in *AWS SDK for Kotlin API reference*.

`DeleteAccessKey`

The following code example shows how to use `DeleteAccessKey`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteKey(
```

```
    userNameVal: String,
    accessKey: String,
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- For API details, see DeleteAccessKey in *AWS SDK for Kotlin API reference*.

`DeleteAccountAlias`

The following code example shows how to use `DeleteAccountAlias`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- For API details, see DeleteAccountAlias in *AWS SDK for Kotlin API reference*.

`DeletePolicy`

The following code example shows how to use `DeletePolicy`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- For API details, see DeletePolicy in *AWS SDK for Kotlin API reference*.

`DeleteUser`

The following code example shows how to use `DeleteUser`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- For API details, see DeleteUser in *AWS SDK for Kotlin API reference*.

`DetachRolePolicy`

The following code example shows how to use `DetachRolePolicy`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
```

```
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
}
```

- For API details, see DetachRolePolicy in *AWS SDK for Kotlin API reference*.

`GetPolicy`

The following code example shows how to use `GetPolicy`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- For API details, see GetPolicy in *AWS SDK for Kotlin API reference*.

`ListAccessKeys`

The following code example shows how to use `ListAccessKeys`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- For API details, see ListAccessKeys in *AWS SDK for Kotlin API reference*.

`ListAccountAliases`

The following code example shows how to use `ListAccountAliases`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listAliases() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- For API details, see ListAccountAliases in *AWS SDK for Kotlin API reference*.

`ListUsers`

The following code example shows how to use `ListUsers`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listAllUsers() {
    IamClient.fromEnvironment { 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}")
            }
        }
    }
}
```

- For API details, see ListUsers in *AWS SDK for Kotlin API reference*.

`UpdateUser`

The following code example shows how to use `UpdateUser`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- For API details, see UpdateUser in *AWS SDK for Kotlin API reference*.

AWS IoT examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS IoT.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello AWS IoT

The following code examples show how to get started using AWS IoT.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import 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.fromEnvironment { 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}")
            }
        }
    }
}
```

- For API details, see listThings in *AWS SDK for Kotlin API reference*.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- Create an AWS IoT Thing.
- Generate a device certificate.
- Update an AWS IoT Thing with Attributes.
- Return a unique endpoint.
- List your AWS IoT certificates.
- Create an AWS IoT shadow.
- Write out state information.
- Creates a rule.
- List your rules.
- Search things using the Thing name.
- Delete an AWS IoT Thing.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import 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.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

Actions

`AttachThingPrincipal`

The following code example shows how to use `AttachThingPrincipal`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- For API details, see AttachThingPrincipal in *AWS SDK for Kotlin API reference*.

`CreateKeysAndCertificate`

The following code example shows how to use `CreateKeysAndCertificate`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { 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
    }
}
```

- For API details, see CreateKeysAndCertificate in *AWS SDK for Kotlin API reference*.

`CreateThing`

The following code example shows how to use `CreateThing`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- For API details, see CreateThing in *AWS SDK for Kotlin API reference*.

`CreateTopicRule`

The following code example shows how to use `CreateTopicRule`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- For API details, see CreateTopicRule in *AWS SDK for Kotlin API reference*.

`DeleteCertificate`

The following code example shows how to use `DeleteCertificate`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
```

```
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- For API details, see DeleteCertificate in *AWS SDK for Kotlin API reference*.

`DeleteThing`

The following code example shows how to use `DeleteThing`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- For API details, see DeleteThing in *AWS SDK for Kotlin API reference*.

`DescribeEndpoint`

The following code example shows how to use `DescribeEndpoint`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { 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
    }
}
```

- For API details, see DescribeEndpoint in *AWS SDK for Kotlin API reference*.

`DescribeThing`

The following code example shows how to use `DescribeThing`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }
```

```
    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- For API details, see DescribeThing in *AWS SDK for Kotlin API reference*.

`DetachThingPrincipal`

The following code example shows how to use `DetachThingPrincipal`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- For API details, see DetachThingPrincipal in *AWS SDK for Kotlin API reference*.

`ListCertificates`

The following code example shows how to use `ListCertificates`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listCertificates() {
    IotClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see ListCertificates in *AWS SDK for Kotlin API reference*.

`SearchIndex`

The following code example shows how to use `SearchIndex`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { 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}") }
        }
    }
}
```

- For API details, see SearchIndex in *AWS SDK for Kotlin API reference*.

`UpdateThing`

The following code example shows how to use `UpdateThing`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- For API details, see UpdateThing in *AWS SDK for Kotlin API reference*.

AWS IoT data examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS IoT data.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Actions

Actions

`GetThingShadow`

The following code example shows how to use `GetThingShadow`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { 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")
    }
}
```

- For API details, see GetThingShadow in *AWS SDK for Kotlin API reference*.

`UpdateThingShadow`

The following code example shows how to use `UpdateThingShadow`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { iotPlaneClient
 ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}
```

- For API details, see [UpdateThingShadow](#) in *AWS SDK for Kotlin API reference*.

AWS IoT FleetWise examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS IoT FleetWise.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello AWS IoT FleetWise

The following code examples show how to get started using AWS IoT FleetWise.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
Before running this 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() {
    listSignalCatalogs()
}

/**
 * Lists the AWS FleetWise Signal Catalogs associated with the current AWS account.
 */
suspend fun listSignalCatalogs() {
    val request = ListSignalCatalogsRequest {
        maxResults = 10
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.listSignalCatalogs(request)
        val summaries = response.summaries

        if (summaries.isNullOrEmpty()) {
            println("No AWS FleetWise Signal Catalogs were found.")
        } else {
            summaries.forEach { summary ->
                with(summary) {
                    println("Catalog Name: $name")
                    println("ARN: $arn")
                    println("Created: $creationTime")
                    println("Last Modified: $lastModificationTime")
                    println("---------------")
                }
```

```
            }
        }
    }
}
```

- For API details, see listSignalCatalogsPaginator in *AWS SDK for Kotlin API reference*.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- Create a collection of standardized signals.
- Create a fleet that represents a group of vehicles.
- Create a model manifest.
- Create a decoder manifest.
- Check the status of the model manifest.
- Check the status of the decoder.
- Create an IoT Thing.
- Create a vehicle.
- Display vehicle details.
- Delete the AWS IoT FleetWise Assets.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run an interactive scenario demonstrating AWS IoT SiteWise features.

```
/**
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
 */
var scanner = Scanner(System.`in`)
val DASHES = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <signalCatalogName> <manifestName> <fleetId> <vecName> <decName>

        Where:
            signalCatalogName     - The name of the Signal Catalog to create (eg,
 catalog30).
            manifestName          - The name of the Vehicle Model (Model Manifest)
 to create (eg, manifest30).
            fleetId               - The ID of the Fleet to create (eg, fleet30).
            vecName               - The name of the Vehicle to create (eg,
 vehicle30).
            decName               - The name of the Decoder Manifest to create (eg,
 decManifest30).

        """.trimIndent()

   if (args.size != 5) {
        println(usage)
        return
   }

   val signalCatalogName = args[0]
   val manifestName = args[1]
   val fleetId = args[2]
   val vecName = args[3]
   val decName = args[4]

   println(
        """
        AWS IoT FleetWise is a managed service that simplifies the
```

```
        process of collecting, organizing, and transmitting vehicle
        data to the cloud in near real-time. Designed for automakers
        and fleet operators, it allows you to define vehicle models,
        specify the exact data you want to collect (such as engine
        temperature, speed, or battery status), and send this data to
        AWS for analysis. By using intelligent data collection
        techniques, IoT FleetWise reduces the volume of data
        transmitted by filtering and transforming it at the edge,
        helping to minimize bandwidth usage and costs.

        At its core, AWS IoT FleetWise helps organizations build
        scalable systems for vehicle data management and analytics,
        supporting a wide variety of vehicles and sensor configurations.
        You can define signal catalogs and decoder manifests that describe
        how raw CAN bus signals are translated into readable data, making
        the platform highly flexible and extensible. This allows
        manufacturers to optimize vehicle performance, improve safety,
        and reduce maintenance costs by gaining real-time visibility
        into fleet operations.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    println(DASHES)
    runScenario(signalCatalogName, fleetId, manifestName, decName, vecName)
}

suspend fun runScenario(signalCatalogName: String, fleetIdVal: String, manifestName:
 String, decName: String, vecName: String) {
    println(DASHES)
    println("1. Creates a collection of standardized signals that can be reused to
 create vehicle models")
    waitForInputToContinue(scanner)
    val signalCatalogArn = createbranchVehicle(signalCatalogName)
    println("The collection ARN is $signalCatalogArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("2. Create a fleet that represents a group of vehicles")
    println(
        """
        Creating an IoT FleetWise fleet allows you to efficiently collect,
        organize, and transfer vehicle data to the cloud, enabling real-time
        insights into vehicle performance and health.
```

```
        It helps reduce data costs by allowing you to filter and prioritize
        only the most relevant vehicle signals, supporting advanced analytics
        and predictive maintenance use cases.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val fleetid = createFleet(signalCatalogArn, fleetIdVal)
    println("The fleet Id is $fleetid")
    waitForInputToContinue(scanner)
    val nodeList = listSignalCatalogNode(signalCatalogName)
    println(DASHES)

    println(DASHES)
    println("3. Create a model manifest")
    println(
        """
        An AWS IoT FleetWise manifest defines the structure and
        relationships of vehicle data. The model manifest specifies
        which signals to collect and how they relate to vehicle systems,
        while the decoder manifest defines how to decode raw vehicle data
        into meaningful signals.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val nodes = listSignalCatalogNode(signalCatalogName)
    val manifestArn = nodes?.let { createModelManifest(manifestName,
signalCatalogArn, it) }
    println("The manifest ARN is $manifestArn")
    println(DASHES)

    println(DASHES)
    println("4. Create a decoder manifest")
    println(
        """
        A decoder manifest in AWS IoT FleetWise defines how raw vehicle
        data (such as CAN signals) should be interpreted and decoded
        into meaningful signals. It acts as a translation layer
        that maps vehicle-specific protocols to standardized data formats
        using decoding rules. This is crucial for extracting usable
        data from different vehicle models, even when their data
        formats vary.
        """.trimIndent(),
    )
```

```
    waitForInputToContinue(scanner)
    val decArn = createDecoderManifest(decName, manifestArn)
    println("The decoder manifest ARN is $decArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("5. Check the status of the model manifest")
    println(
        """
        The model manifest must be in an ACTIVE state before it can be used
        to create or update a vehicle.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    updateModelManifest(manifestName)
    waitForModelManifestActive(manifestName)
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("6. Check the status of the decoder")
    println(
        """
        The decoder manifest must be in an ACTIVE state before it can be used
        to create or update a vehicle.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    updateDecoderManifest(decName)
    waitForDecoderManifestActive(decName)
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("7. Create an IoT Thing")
    println(
        """
        AWS IoT FleetWise expects an existing AWS IoT Thing with the same
        name as the vehicle name you are passing to createVehicle method.
        Before calling createVehicle(), you must create an AWS IoT Thing
        with the same name using the AWS IoT Core service.
        """.trimIndent(),
    )
```

```
    waitForInputToContinue(scanner)
    createThingIfNotExist(vecName)
    println(DASHES)

    println(DASHES)
    println("8. Create a vehicle")
    println(
        """
        Creating a vehicle in AWS IoT FleetWise allows you to digitally
        represent and manage a physical vehicle within the AWS ecosystem.
        This enables efficient ingestion, transformation, and transmission
        of vehicle telemetry data to the cloud for analysis.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    createVehicle(vecName, manifestArn, decArn)
    println(DASHES)

    println(DASHES)
    println("9. Display vehicle details")
    waitForInputToContinue(scanner)
    getVehicleDetails(vecName)
    waitForInputToContinue(scanner)
    println(DASHES)
    println(DASHES)
    println("10. Delete the AWS IoT Fleetwise Assets")
    println("Would you like to delete the IoT Fleetwise Assets? (y/n)")
    val delAns = scanner.nextLine().trim()
    if (delAns.equals("y", ignoreCase = true)) {
        deleteVehicle(vecName)
        deleteDecoderManifest(decName)
        deleteModelManifest(manifestName)
        deleteFleet(fleetid)
        deleteSignalCatalog(signalCatalogName)
    }

    println(DASHES)
    println(
        """
        Thank you for checking out the AWS IoT Fleetwise Service Use demo. We hope
you
        learned something new, or got some inspiration for your own apps today.
        For more AWS code examples, have a look at:
        https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-library.html
```

```
        """.trimIndent(),
    )
    println(DASHES)
}

suspend fun deleteVehicle(vecName: String) {
    val request = DeleteVehicleRequest {
        vehicleName = vecName
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteVehicle(request)
        println("Vehicle $vecName was deleted successfully.")
    }
}

suspend fun getVehicleDetails(vehicleNameVal: String) {
    val request = GetVehicleRequest {
        vehicleName = vehicleNameVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.getVehicle(request)
        val details = mapOf(
            "vehicleName" to response.vehicleName,
            "arn" to response.arn,
            "modelManifestArn" to response.modelManifestArn,
            "decoderManifestArn" to response.decoderManifestArn,
            "attributes" to response.attributes.toString(),
            "creationTime" to response.creationTime.toString(),
            "lastModificationTime" to response.lastModificationTime.toString(),
        )

        println("Vehicle Details:")
        for ((key, value) in details) {
            println("• %-20s : %s".format(key, value))
        }
    }
}

suspend fun createVehicle(vecName: String, manifestArn: String?, decArn: String) {
    val request = CreateVehicleRequest {
```

```
        vehicleName = vecName
        modelManifestArn = manifestArn
        decoderManifestArn = decArn
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.createVehicle(request)
        println("Vehicle $vecName was created successfully.")
    }
}

/**
 * Creates an IoT Thing if it does not already exist.
 *
 * @param vecName the name of the IoT Thing to create
 */
suspend fun createThingIfNotExist(vecName: String) {
    val request = CreateThingRequest {
        thingName = vecName
    }

    IotClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.createThing(request)
        println("The $vecName IoT Thing was successfully created")
    }
}

suspend fun updateDecoderManifest(nameVal: String) {
    val request = UpdateDecoderManifestRequest {
        name = nameVal
        status = ManifestStatus.Active
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.updateDecoderManifest(request)
        println("$nameVal was successfully updated")
    }
}

/**
 * Waits for the specified model manifest to become active.
 *
 * @param decNameVal the name of the model manifest to wait for
```

```
 */
suspend fun waitForDecoderManifestActive(decNameVal: String) {
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        while (true) {
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
                val request = GetDecoderManifestRequest {
                    name = decNameVal
                }

                val response = fleetwiseClient.getDecoderManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft

                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
                        return
                    }

                    ManifestStatus.Invalid -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: INVALID #\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
                    }

                    else -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
        }
    }
}

/**
```

```
 * Waits for the specified model manifest to become active.
 *
 * @param manifestName the name of the model manifest to wait for
 */
suspend fun waitForModelManifestActive(manifestNameVal: String) {
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        while (true) {
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
                val request = GetModelManifestRequest {
                    name = manifestNameVal
                }

                val response = fleetwiseClient.getModelManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft

                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
                        return
                    }

                    ManifestStatus.Invalid -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: INVALID #
\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
                    }

                    else -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
        }
```

```
    }
}

/**
 * Updates the model manifest.
 *
 * @param nameVal the name of the model manifest to update
 */
suspend fun updateModelManifest(nameVal: String) {
    val request = UpdateModelManifestRequest {
        name = nameVal
        status = ManifestStatus.Active
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.updateModelManifest(request)
        println("$nameVal was successfully updated")
    }
}

suspend fun deleteDecoderManifest(nameVal: String) {
    val request = DeleteDecoderManifestRequest {
        name = nameVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteDecoderManifest(request)
        println("$nameVal was successfully deleted")
    }
}

/**
 * Creates a new decoder manifest.
 *
 * @param decName             the name of the decoder manifest
 * @param modelManifestArnVal the ARN of the model manifest
 * @return the ARN of the decoder manifest
 */
suspend fun createDecoderManifest(decName: String, modelManifestArnVal: String?):
 String {
    val interfaceIdVal = "can0"

    val canInter = CanInterface {
```

```
        name = "canInterface0"
        protocolName = "CAN"
        protocolVersion = "1.0"
    }

    val networkInterface = NetworkInterface {
        interfaceId = interfaceIdVal
        type = NetworkInterfaceType.CanInterface
        canInterface = canInter
    }

    val carRpmSig = CanSignal {
        messageId = 100
        isBigEndian = false
        isSigned = false
        startBit = 16
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val carSpeedSig = CanSignal {
        messageId = 101
        isBigEndian = false
        isSigned = false
        startBit = 0
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val engineRpmDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
        canSignal = carRpmSig
    }

    val vehicleSpeedDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
        canSignal = carSpeedSig
    }
```

```
    val request = CreateDecoderManifestRequest {
        name = decName
        modelManifestArn = modelManifestArnVal
        networkInterfaces = listOf(networkInterface)
        signalDecoders = listOf(engineRpmDecoder, vehicleSpeedDecoder)
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createDecoderManifest(request)
        return response.arn
    }
}

/**
 * Deletes a signal catalog.
 *
 * @param name the name of the signal catalog to delete
 */
suspend fun deleteSignalCatalog(catName: String) {
    val request = DeleteSignalCatalogRequest {
        name = catName
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteSignalCatalog(request)
        println(" $catName was successfully deleted")
    }
}

/**
 * Deletes a fleet based on the provided fleet ID.
 *
 * @param fleetId the ID of the fleet to be deleted
 */
suspend fun deleteFleet(fleetIdVal: String) {
    val request = DeleteFleetRequest {
        fleetId = fleetIdVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteFleet(request)
```

```
        println(" $fleetIdVal was successfully deleted")
    }
}

/**
 * Deletes a model manifest.
 *
 * @param nameVal the name of the model manifest to delete
 */
suspend fun deleteModelManifest(nameVal: String) {
    val request = DeleteModelManifestRequest {
        name = nameVal
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteModelManifest(request)
        println(" $nameVal was successfully deleted")
    }
}

/**
 * Creates a model manifest.
 *
 * @param name              the name of the model manifest to create
 * @param signalCatalogArn  the Amazon Resource Name (ARN) of the signal catalog
 * @param nodes             a list of nodes to include in the model manifest
 * @return a {@link CompletableFuture} that completes with the ARN of the created
 model manifest
 */
suspend fun createModelManifest(nameVal: String, signalCatalogArnVal: String,
 nodesList: List<Node>): String {
    val fqnList: List<String> = nodesList.map { node ->
        when (node) {
            is Node.Sensor -> node.asSensor().fullyQualifiedName
            is Node.Branch -> node.asBranch().fullyQualifiedName
            else -> throw RuntimeException("Unsupported node type")
        }
    }

    val request = CreateModelManifestRequest {
        name = nameVal
        signalCatalogArn = signalCatalogArnVal
        nodes = fqnList
    }
```

```
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createModelManifest(request)
        return response.arn
    }
}

/**
 * Lists the signal catalog nodes asynchronously.
 *
 * @param signalCatalogName the name of the signal catalog
 * @return a CompletableFuture that, when completed, contains a list of nodes in the
 specified signal catalog
 * @throws CompletionException if an exception occurs during the asynchronous
 operation
 */
suspend fun listSignalCatalogNode(signalCatalogName: String): List<Node>? {
    val request = ListSignalCatalogNodesRequest {
        name = signalCatalogName
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.listSignalCatalogNodes(request)
        return response.nodes
    }
}

/**
 * Creates a new fleet.
 *
 * @param catARN the Amazon Resource Name (ARN) of the signal catalog to associate
 with the fleet
 * @param fleetId the unique identifier for the fleet
 * @return the ID of the created fleet
 */
suspend fun createFleet(catARN: String, fleetIdVal: String): String {
    val fleetRequest = CreateFleetRequest {
        fleetId = fleetIdVal
        signalCatalogArn = catARN
        description = "Built using the AWS For Kotlin"
    }
```

```
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createFleet(fleetRequest)
        return response.id
    }
}

/**
 * Creates a signal catalog.
 *
 * @param signalCatalogName the name of the signal catalog to create the branch
 vehicle in
 * @return the ARN (Amazon Resource Name) of the created signal catalog
 */
suspend fun createbranchVehicle(signalCatalogName: String): String {
    delay(2000) // Wait for 2 seconds
    val branchVehicle = Branch {
        fullyQualifiedName = "Vehicle"
        description = "Root branch"
    }

    val branchPowertrain = Branch {
        fullyQualifiedName = "Vehicle.Powertrain"
        description = "Powertrain branch"
    }

    val sensorRPM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        description = "Engine RPM"
        dataType = NodeDataType.Double
        unit = "rpm"
    }

    val sensorKM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        description = "Vehicle Speed"
        dataType = NodeDataType.Double
        unit = "km/h"
    }

    // Wrap each specific node type (Branch and Sensor) into the sealed Node class
    // so they can be included in the CreateSignalCatalogRequest.
    val myNodes = listOf(
        Node.Branch(branchVehicle),
```

```
        Node.Branch(branchPowertrain),
        Node.Sensor(sensorRPM),
        Node.Sensor(sensorKM),
    )

    val request = CreateSignalCatalogRequest {
        name = signalCatalogName
        nodes = myNodes
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createSignalCatalog(request)
        return response.arn
    }
}

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 {
            println("Invalid input. Please try again.")
        }
    }
}
```

Actions

`createDecoderManifest`

The following code example shows how to use `createDecoderManifest`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Creates a new decoder manifest.
 *
 * @param decName             the name of the decoder manifest
 * @param modelManifestArnVal the ARN of the model manifest
 * @return the ARN of the decoder manifest
 */
suspend fun createDecoderManifest(decName: String, modelManifestArnVal: String?):
 String {
    val interfaceIdVal = "can0"

    val canInter = CanInterface {
        name = "canInterface0"
        protocolName = "CAN"
        protocolVersion = "1.0"
    }

    val networkInterface = NetworkInterface {
        interfaceId = interfaceIdVal
        type = NetworkInterfaceType.CanInterface
        canInterface = canInter
    }

    val carRpmSig = CanSignal {
        messageId = 100
        isBigEndian = false
        isSigned = false
        startBit = 16
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val carSpeedSig = CanSignal {
```

```
        messageId = 101
        isBigEndian = false
        isSigned = false
        startBit = 0
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val engineRpmDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
        canSignal = carRpmSig
    }

    val vehicleSpeedDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
        canSignal = carSpeedSig
    }

    val request = CreateDecoderManifestRequest {
        name = decName
        modelManifestArn = modelManifestArnVal
        networkInterfaces = listOf(networkInterface)
        signalDecoders = listOf(engineRpmDecoder, vehicleSpeedDecoder)
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createDecoderManifest(request)
        return response.arn
    }
}
```

- For API details, see [createDecoderManifest](#) in *AWS SDK for Kotlin API reference*.

`createFleet`

The following code example shows how to use `createFleet`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Creates a new fleet.
 *
 * @param catARN the Amazon Resource Name (ARN) of the signal catalog to associate
 with the fleet
 * @param fleetId the unique identifier for the fleet
 * @return the ID of the created fleet
 */
suspend fun createFleet(catARN: String, fleetIdVal: String): String {
    val fleetRequest = CreateFleetRequest {
        fleetId = fleetIdVal
        signalCatalogArn = catARN
        description = "Built using the AWS For Kotlin"
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createFleet(fleetRequest)
        return response.id
    }
}
```

- For API details, see createFleet in *AWS SDK for Kotlin API reference*.

`createModelManifest`

The following code example shows how to use `createModelManifest`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Creates a model manifest.
 *
 * @param name              the name of the model manifest to create
 * @param signalCatalogArn  the Amazon Resource Name (ARN) of the signal catalog
 * @param nodes             a list of nodes to include in the model manifest
 * @return a {@link CompletableFuture} that completes with the ARN of the created
 model manifest
 */
suspend fun createModelManifest(nameVal: String, signalCatalogArnVal: String,
 nodesList: List<Node>): String {
    val fqnList: List<String> = nodesList.map { node ->
        when (node) {
            is Node.Sensor -> node.asSensor().fullyQualifiedName
            is Node.Branch -> node.asBranch().fullyQualifiedName
            else -> throw RuntimeException("Unsupported node type")
        }
    }

    val request = CreateModelManifestRequest {
        name = nameVal
        signalCatalogArn = signalCatalogArnVal
        nodes = fqnList
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createModelManifest(request)
        return response.arn
    }
}
```

- For API details, see createModelManifest in *AWS SDK for Kotlin API reference*.

`createSignalCatalog`

The following code example shows how to use `createSignalCatalog`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Creates a signal catalog.
 *
 * @param signalCatalogName the name of the signal catalog to create the branch
 vehicle in
 * @return the ARN (Amazon Resource Name) of the created signal catalog
 */
suspend fun createbranchVehicle(signalCatalogName: String): String {
    delay(2000) // Wait for 2 seconds
    val branchVehicle = Branch {
        fullyQualifiedName = "Vehicle"
        description = "Root branch"
    }

    val branchPowertrain = Branch {
        fullyQualifiedName = "Vehicle.Powertrain"
        description = "Powertrain branch"
    }

    val sensorRPM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        description = "Engine RPM"
        dataType = NodeDataType.Double
        unit = "rpm"
    }

    val sensorKM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        description = "Vehicle Speed"
        dataType = NodeDataType.Double
        unit = "km/h"
```

```
    }

    // Wrap each specific node type (Branch and Sensor) into the sealed Node class
    // so they can be included in the CreateSignalCatalogRequest.
    val myNodes = listOf(
        Node.Branch(branchVehicle),
        Node.Branch(branchPowertrain),
        Node.Sensor(sensorRPM),
        Node.Sensor(sensorKM),
    )

    val request = CreateSignalCatalogRequest {
        name = signalCatalogName
        nodes = myNodes
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createSignalCatalog(request)
        return response.arn
    }
}
```

- For API details, see [createSignalCatalog](#) in *AWS SDK for Kotlin API reference*.

`createVehicle`

The following code example shows how to use `createVehicle`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createVehicle(vecName: String, manifestArn: String?, decArn: String) {
    val request = CreateVehicleRequest {
        vehicleName = vecName
```

```
        modelManifestArn = manifestArn
        decoderManifestArn = decArn
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.createVehicle(request)
        println("Vehicle $vecName was created successfully.")
    }
}
```

- For API details, see createVehicle in *AWS SDK for Kotlin API reference*.

`deleteDecoderManifest`

The following code example shows how to use `deleteDecoderManifest`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteDecoderManifest(nameVal: String) {
    val request = DeleteDecoderManifestRequest {
        name = nameVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteDecoderManifest(request)
        println("$nameVal was successfully deleted")
    }
}
```

- For API details, see deleteDecoderManifest in *AWS SDK for Kotlin API reference*.

`deleteFleet`

The following code example shows how to use `deleteFleet`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Deletes a fleet based on the provided fleet ID.
 *
 * @param fleetId the ID of the fleet to be deleted
 */
suspend fun deleteFleet(fleetIdVal: String) {
    val request = DeleteFleetRequest {
        fleetId = fleetIdVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteFleet(request)
        println(" $fleetIdVal was successfully deleted")
    }
}
```

- For API details, see deleteFleet in *AWS SDK for Kotlin API reference*.

`deleteModelManifest`

The following code example shows how to use `deleteModelManifest`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Deletes a model manifest.
 *
 * @param nameVal the name of the model manifest to delete
 */
suspend fun deleteModelManifest(nameVal: String) {
    val request = DeleteModelManifestRequest {
        name = nameVal
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteModelManifest(request)
        println(" $nameVal was successfully deleted")
    }
}
```

- For API details, see deleteModelManifest in *AWS SDK for Kotlin API reference*.

`deleteSignalCatalog`

The following code example shows how to use `deleteSignalCatalog`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
```

```
 * Deletes a signal catalog.
 *
 * @param name the name of the signal catalog to delete
 */
suspend fun deleteSignalCatalog(catName: String) {
    val request = DeleteSignalCatalogRequest {
        name = catName
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteSignalCatalog(request)
        println(" $catName was successfully deleted")
    }
}
```

- For API details, see deleteSignalCatalog in *AWS SDK for Kotlin API reference*.

`deleteVehicle`

The following code example shows how to use `deleteVehicle`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteVehicle(vecName: String) {
    val request = DeleteVehicleRequest {
        vehicleName = vecName
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteVehicle(request)
        println("Vehicle $vecName was deleted successfully.")
    }
}
```

- For API details, see deleteVehicle in *AWS SDK for Kotlin API reference*.

`getDecoderManifest`

The following code example shows how to use `getDecoderManifest`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Waits for the specified model manifest to become active.
 *
 * @param decNameVal the name of the model manifest to wait for
 */
suspend fun waitForDecoderManifestActive(decNameVal: String) {
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        while (true) {
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
                val request = GetDecoderManifestRequest {
                    name = decNameVal
                }

                val response = fleetwiseClient.getDecoderManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft

                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
```

```
                        return
                    }

                    ManifestStatus.Invalid -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: INVALID #\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
                    }

                    else -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
        }
    }
}
```

- For API details, see [getDecoderManifest](#) in *AWS SDK for Kotlin API reference*.

`getModelManifest`

The following code example shows how to use `getModelManifest`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
 * Waits for the specified model manifest to become active.
 *
 * @param manifestName the name of the model manifest to wait for
 */
suspend fun waitForModelManifestActive(manifestNameVal: String) {
```

```
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        while (true) {
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
                val request = GetModelManifestRequest {
                    name = manifestNameVal
                }

                val response = fleetwiseClient.getModelManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft

                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
                        return
                    }

                    ManifestStatus.Invalid -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: INVALID #
\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
                    }

                    else -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
        }
    }
}
```

- For API details, see [getModelManifest](#) in *AWS SDK for Kotlin API reference*.

`getVehicle`

The following code example shows how to use `getVehicle`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getVehicleDetails(vehicleNameVal: String) {
    val request = GetVehicleRequest {
        vehicleName = vehicleNameVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.getVehicle(request)
        val details = mapOf(
            "vehicleName" to response.vehicleName,
            "arn" to response.arn,
            "modelManifestArn" to response.modelManifestArn,
            "decoderManifestArn" to response.decoderManifestArn,
            "attributes" to response.attributes.toString(),
            "creationTime" to response.creationTime.toString(),
            "lastModificationTime" to response.lastModificationTime.toString(),
        )

        println("Vehicle Details:")
        for ((key, value) in details) {
            println("• %-20s : %s".format(key, value))
        }
    }
}
```

- For API details, see getVehicle in *AWS SDK for Kotlin API reference*.

`listSignalCatalogNodes`

The following code example shows how to use `listSignalCatalogNodes`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Lists the signal catalog nodes asynchronously.
 *
 * @param signalCatalogName the name of the signal catalog
 * @return a CompletableFuture that, when completed, contains a list of nodes in the
 specified signal catalog
 * @throws CompletionException if an exception occurs during the asynchronous
 operation
 */
suspend fun listSignalCatalogNode(signalCatalogName: String): List<Node>? {
    val request = ListSignalCatalogNodesRequest {
        name = signalCatalogName
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.listSignalCatalogNodes(request)
        return response.nodes
    }
}
```

- For API details, see listSignalCatalogNodes in *AWS SDK for Kotlin API reference*.

`updateDecoderManifest`

The following code example shows how to use `updateDecoderManifest`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun updateDecoderManifest(nameVal: String) {
    val request = UpdateDecoderManifestRequest {
        name = nameVal
        status = ManifestStatus.Active
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.updateDecoderManifest(request)
        println("$nameVal was successfully updated")
    }
}
```

- For API details, see updateDecoderManifest in *AWS SDK for Kotlin API reference*.

`updateModelManifest`

The following code example shows how to use `updateModelManifest`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Updates the model manifest.
 *
 * @param nameVal the name of the model manifest to update
 */
```

```
suspend fun updateModelManifest(nameVal: String) {
    val request = UpdateModelManifestRequest {
        name = nameVal
        status = ManifestStatus.Active
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.updateModelManifest(request)
        println("$nameVal was successfully updated")
    }
}
```

- For API details, see updateModelManifest in *AWS SDK for Kotlin API reference*.

Amazon Keyspaces examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Keyspaces.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon Keyspaces

The following code examples show how to get started using Amazon Keyspaces.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
Before running this 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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- For API details, see ListKeyspaces in *AWS SDK for Kotlin API reference*.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- Create a keyspace and table. The table schema holds movie data and has point-in-time recovery enabled.
- Connect to the keyspace using a secure TLS connection with SigV4 authentication.
- Query the table. Add, retrieve, and update movie data.
- Update the table. Add a column to track watched movies.
- Restore the table to its previous state and clean up resources.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 Before running this 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Actions

`CreateKeyspace`

The following code example shows how to use `CreateKeyspace`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- For API details, see CreateKeyspace in *AWS SDK for Kotlin API reference*.

`CreateTable`

The following code example shows how to use `CreateTable`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}
```

- For API details, see CreateTable in *AWS SDK for Kotlin API reference*.

`DeleteKeyspace`

The following code example shows how to use `DeleteKeyspace`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- For API details, see DeleteKeyspace in *AWS SDK for Kotlin API reference*.

`DeleteTable`

The following code example shows how to use `DeleteTable`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
```

```
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- For API details, see DeleteTable in *AWS SDK for Kotlin API reference*.

`GetKeyspace`

The following code example shows how to use `GetKeyspace`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}
```

- For API details, see GetKeyspace in *AWS SDK for Kotlin API reference*.

`GetTable`

The following code example shows how to use `GetTable`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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}")
            }
        }
    }
```

```
}
```

- For API details, see GetTable in *AWS SDK for Kotlin API reference*.

`ListKeyspaces`

The following code example shows how to use `ListKeyspaces`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}
```

- For API details, see ListKeyspaces in *AWS SDK for Kotlin API reference*.

`ListTables`

The following code example shows how to use `ListTables`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { 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}")
            }
    }
}
```

- For API details, see ListTables in *AWS SDK for Kotlin API reference*.

`RestoreTable`

The following code example shows how to use `RestoreTable`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- For API details, see RestoreTable in *AWS SDK for Kotlin API reference*.

`UpdateTable`

The following code example shows how to use `UpdateTable`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}
```

- For API details, see UpdateTable in *AWS SDK for Kotlin API reference*.

AWS KMS examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS KMS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Actions

Actions

`CreateAlias`

The following code example shows how to use `CreateAlias`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- For API details, see CreateAlias in *AWS SDK for Kotlin API reference*.

`CreateGrant`

The following code example shows how to use `CreateGrant`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

- For API details, see CreateGrant in *AWS SDK for Kotlin API reference*.

`CreateKey`

The following code example shows how to use `CreateKey`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
```

```
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient.fromEnvironment { 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
    }
}
```

- For API details, see CreateKey in *AWS SDK for Kotlin API reference*.

`Decrypt`

The following code example shows how to use `Decrypt`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- For API details, see Decrypt in *AWS SDK for Kotlin API reference*.

`DescribeKey`

The following code example shows how to use `DescribeKey`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }
```

```
    KmsClient.fromEnvironment { 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}")
    }
}
```

- For API details, see DescribeKey in *AWS SDK for Kotlin API reference*.

`DisableKey`

The following code example shows how to use `DisableKey`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- For API details, see DisableKey in *AWS SDK for Kotlin API reference*.

`EnableKey`

The following code example shows how to use `EnableKey`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- For API details, see EnableKey in *AWS SDK for Kotlin API reference*.

`Encrypt`

The following code example shows how to use `Encrypt`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- For API details, see Encrypt in *AWS SDK for Kotlin API reference*.

`ListAliases`

The following code example shows how to use `ListAliases`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- For API details, see ListAliases in *AWS SDK for Kotlin API reference*.

`ListGrants`

The following code example shows how to use `ListGrants`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
```

```
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- For API details, see ListGrants in *AWS SDK for Kotlin API reference*.

`ListKeys`

The following code example shows how to use `ListKeys`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listAllKeys() {
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see ListKeys in *AWS SDK for Kotlin API reference*.

Lambda examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Lambda.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Basics
- Actions
- Scenarios

Basics

Learn the basics

The following code example shows how to:

- Create an IAM role and Lambda function, then upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure with an environment variable.
- Invoke the function with new parameters and get results. Display the returned execution log.
- List the functions for your account, then clean up resources.

For more information, see Create a Lambda function with the console.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.Java17
        }

    // Create a Lambda function using a waiter
    LambdaClient { region = "us-east-1" }.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-east-1" }.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-east-1" }.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-east-1" }.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-east-1" }.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.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.updateFunctionConfiguration(configurationRequest)
    }
}

suspend fun delFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Actions

`CreateFunction`

The following code example shows how to use `CreateFunction`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn
    }
}
```

- For API details, see CreateFunction in *AWS SDK for Kotlin API reference*.

`DeleteFunction`

The following code example shows how to use `DeleteFunction`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- For API details, see DeleteFunction in *AWS SDK for Kotlin API reference*.

`Invoke`

The following code example shows how to use `Invoke`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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}")
    }
}
```

- For API details, see Invoke in *AWS SDK for Kotlin API reference*.

Scenarios

Create a serverless application to manage photos

The following code example shows how to create a serverless application that lets users manage photos using labels.

SDK for Kotlin

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition

- Amazon S3
- Amazon SNS

Amazon Location examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Location.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon Location Service

The following code examples show how to get started using Amazon Location Service.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
Before running this 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 need to create a collection using the AWS Management
console. For information, see the following documentation.

https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html
```

```
 */
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:
            <colletionName>

        Where:
            colletionName - The Amazon location collection name.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }
    val colletionName = args[0]
    listGeofences(colletionName)
}

/**
 * Lists the geofences for the specified collection name.
 *
 * @param collectionName the name of the geofence collection
 */
suspend fun listGeofences(collectionName: String) {
    val request = ListGeofencesRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.listGeofences(request)
        val geofences = response.entries
        if (geofences.isNullOrEmpty()) {
            println("No Geofences found")
        } else {
            geofences.forEach { geofence ->
                println("Geofence ID: ${geofence.geofenceId}")
            }
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - ListGeofenceCollections
  - ListGeofences

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- Create an Amazon Location map.
- Create an Amazon Location API key.
- Display Map URL.
- Create a geofence collection.
- Store a geofence geometry.
- Create a tracker resource.
- Update the position of a device.
- Retrieve the most recent position update for a specified device.
- Create a route calculator.
- Determine the distance between Seattle and Vancouver.
- Use Amazon Location higher level APIs.
- Delete the Amazon Location Assets.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
Before running this 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 scanner = Scanner(System.`in`)
val DASHES = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

        Where:
            mapName - The name of the map to create (e.g., "AWSMap").
            keyName - The name of the API key to create (e.g., "AWSApiKey").
            collectionName - The name of the geofence collection (e.g.,
 "AWSLocationCollection").
            geoId - The geographic identifier used for the geofence or map (e.g.,
 "geoId").
            trackerName - The name of the tracker (e.g., "geoTracker").
            calculatorName - The name of the route calculator (e.g.,
 "AWSRouteCalc").
            deviceId - The ID of the device (e.g., "iPhone-112356").
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(0)
    }

    val mapName = args[0]
    val keyName = args[1]
    val collectionName = args[2]
    val geoId = args[3]
    val trackerName = args[4]
    val calculatorName = args[5]
    val deviceId = args[6]

    println(
```

```
        """
    AWS Location Service is a fully managed service offered by Amazon Web Services
(AWS) that
    provides location-based services for developers. This service simplifies
    the integration of location-based features into applications, making it
    easier to build and deploy location-aware applications.

    The AWS Location Service offers a range of location-based services,
    including:

    - Maps: The service provides access to high-quality maps, satellite imagery,
      and geospatial data from various providers, allowing developers to
      easily embed maps into their applications.

    - Tracking: The Location Service enables real-time tracking of mobile devices,
      assets, or other entities, allowing developers to build applications
      that can monitor the location of people, vehicles, or other objects.

    - Geocoding: The service provides the ability to convert addresses or
      location names into geographic coordinates (latitude and longitude),
      and vice versa, enabling developers to integrate location-based search
      and routing functionality into their applications.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println(DASHES)
    println("1. Create an AWS Location Service map")
    println(
        """
        An AWS Location map can enhance the user experience of your
        application by providing accurate and personalized location-based
        features. For example, you could use the geocoding capabilities to
        allow users to search for and locate businesses, landmarks, or
        other points of interest within a specific region.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    val mapArn = createMap(mapName)
    println("The Map ARN is: $mapArn")
    waitForInputToContinue(scanner)
    println(DASHES)
```

```
    waitForInputToContinue(scanner)
    println("2. Create an AWS Location API key")
    println(
        """
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.

        """.trimIndent(),
    )
    val keyArn = createKey(keyName, mapArn)
    println("The Key ARN is: $keyArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("3. Display Map URL")
    println(
        """
        In order to get the MAP URL, you need to get the API Key value.
        You can get the key value using the AWS Management Console under
        Location Services. This operation cannot be completed using the
        AWS SDK. For more information about getting the key value, see
        the AWS Location Documentation.
        """.trimIndent(),
    )
    val mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/$mapName/tiles/{z}/
{x}/{y}?key={KeyValue}"
    println("Embed this URL in your Web app: $mapUrl")
    println("")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("4. Create a geofence collection, which manages and stores geofences.")
    waitForInputToContinue(scanner)
    val collectionArn: String =
        createGeofenceCollection(collectionName)
    println("The geofence collection was successfully created: $collectionArn")
    waitForInputToContinue(scanner)
```

```
    println(DASHES)
    println("5. Store a geofence geometry in a given geofence collection.")
    println(
        """
        An AWS Location geofence is a virtual boundary that defines a geographic
area
        on a map. It is a useful feature for tracking the location of
        assets or monitoring the movement of objects within a specific region.

        To define a geofence, you need to specify the coordinates of a
        polygon that represents the area of interest. The polygon must be
        defined in a counter-clockwise direction, meaning that the points of
        the polygon must be listed in a counter-clockwise order.

        This is a requirement for the AWS Location service to correctly
        interpret the geofence and ensure that the location data is
        accurately processed within the defined area.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    putGeofence(collectionName, geoId)
    println("Successfully created geofence: $geoId")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("6. Create a tracker resource which lets you retrieve current and
historical location of devices.")
    waitForInputToContinue(scanner)
    val trackerArn: String = createTracker(trackerName)
    println("Successfully created tracker. ARN: $trackerArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("7. Update the position of a device in the location tracking system.")
    println(
        """
        The AWS location service does not enforce a strict format for deviceId, but
it must:
            - Be a string (case-sensitive).
            - Be 1–100 characters long.
            - Contain only:
```

```
            - Alphanumeric characters (A-Z, a-z, 0-9)
            - Underscores (_)
            - Hyphens (-)
            - Be the same ID used when sending and retrieving positions.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    updateDevicePosition(trackerName, deviceId)
    println("$deviceId was successfully updated in the location tracking system.")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("8. Retrieve the most recent position update for a specified device.")
    waitForInputToContinue(scanner)
    val response = getDevicePosition(trackerName, deviceId)
    println("Successfully fetched device position: ${response.position}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("9. Create a route calculator.")
    waitForInputToContinue(scanner)
    val routeResponse = createRouteCalculator(calculatorName)
    println("Route calculator created successfully: ${routeResponse.calculatorArn}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("10. Determine the distance in kilometers between Seattle and Vancouver
using the route calculator.")
    waitForInputToContinue(scanner)
    val responseDis = calcDistance(calculatorName)
    println("Successfully calculated route. The distance in kilometers is
${responseDis.summary?.distance}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("11. Use the GeoPlacesClient to perform additional operations.")
    println(
        """
```

```
        This scenario will show use of the GeoPlacesClient that enables
        location search and geocoding capabilities for your applications.

        We are going to use this client to perform these AWS Location tasks:
            - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
            - Place Search (searchText): Finds places based on search queries.
            - Nearby Search (searchNearby): Finds places near a specific location.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println("First we will perform a Reverse Geocoding operation")
    waitForInputToContinue(scanner)
    reverseGeocode()

    println("Now we are going to perform a text search using coffee shop.")
    waitForInputToContinue(scanner)
    searchText("coffee shop")
    waitForInputToContinue(scanner)

    println("Now we are going to perform a nearby Search.")
    waitForInputToContinue(scanner)
    searchNearby()
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS Location Services resources.")
    println("Would you like to delete the AWS Location Services resources? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        deleteMap(mapName)
        deleteKey(keyName)
        deleteGeofenceCollection(collectionName)
        deleteTracker(trackerName)
        deleteRouteCalculator(calculatorName)
    } else {
        println("The AWS resources will not be deleted.")
    }
    waitForInputToContinue(scanner)
    println(DASHES)
```

```
    println(DASHES)
    println(" This concludes the AWS Location Service scenario.")
    println(DASHES)
}

/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}


/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}


/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence collection
 has been deleted
 */
```

```
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}
```

```
/**
 * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
 * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
 * The results are processed and printed once the search completes successfully.
 */
suspend fun searchNearby() {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)

    // Set up the request for searching nearby places.
    val request = SearchNearbyRequest {
        this.queryPosition = queryPosition
        this.queryRadius = 1000L
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchNearby(request)

        // Process the response and print the results.
        response.resultItems?.forEach { result ->
            println("Title: ${result.title}")
            println("Address: ${result.address?.label}")
            println("Distance: ${result.distance} meters")
            println("-------------------------")
        }
    }
}


/**
 * Searches for a place using the provided search query and prints the detailed
 information of the first result.
 *
 * @param searchQuery the search query to be used for the place search (ex, coffee
 shop)
 */
suspend fun searchText(searchQuery: String) {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)
```

```
    val request = SearchTextRequest {
        this.queryText = searchQuery
        this.biasPosition = queryPosition
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchText(request)

        response.resultItems?.firstOrNull()?.let { result ->
            val placeId = result.placeId // Get Place ID
            println("Found Place with id: $placeId")

            // Fetch detailed info using getPlace.
            val getPlaceRequest = GetPlaceRequest {
                this.placeId = placeId
            }

            val placeResponse = client.getPlace(getPlaceRequest)

            // Print detailed place information.
            println("Detailed Place Information:")
            println("Title: ${placeResponse.title}")
            println("Address: ${placeResponse.address?.label}")

            // Print each food type (if any).
            placeResponse.foodTypes?.takeIf { it.isNotEmpty() }?.let {
                println("Food Types:")
                it.forEach { foodType ->
                    println("  - $foodType")
                }
            } ?: run {
                println("No food types available.")
            }

            println("-------------------------")
        }
    }
}

/**
 * Performs reverse geocoding using the AWS Geo Places API.
 * Reverse geocoding is the process of converting geographic coordinates (latitude
 and longitude) to a human-readable address.
```

```
 * This method uses the latitude and longitude of San Francisco as the input, and
 prints the resulting address.
 */
suspend fun reverseGeocode() {
    val latitude = 37.7749
    val longitude = -122.4194
    println("Use latitude 37.7749 and longitude -122.4194")

    // AWS expects [longitude, latitude].
    val queryPosition = listOf(longitude, latitude)
    val request = ReverseGeocodeRequest {
        this.queryPosition = queryPosition
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.reverseGeocode(request)
        response.resultItems?.forEach { result ->
            println("The address is: ${result.address?.label}")
        }
    }
}


/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters, Kilometers,
 Miles
    }
```

```
    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}

/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}

/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}

/**
 * Updates the position of a device in the location tracking system.
```

```
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude, latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}

/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the Amazon
 Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
```

```
    }
}

/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}

/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }
```

```
    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}

/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}

/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }

    val mapRequest = CreateMapRequest {
```

```
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}

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 {
            println("Invalid input. Please try again.")
        }
    }
}
```

Actions

`BatchUpdateDevicePosition`

The following code example shows how to use `BatchUpdateDevicePosition`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude, latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}
```

- For API details, see BatchUpdateDevicePosition in *AWS SDK for Kotlin API reference*.

`CalculateRoute`

The following code example shows how to use `CalculateRoute`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters, Kilometers,
 Miles
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}
```

- For API details, see CalculateRoute in *AWS SDK for Kotlin API reference*.

`CreateGeofenceCollection`

The following code example shows how to use `CreateGeofenceCollection`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}
```

- For API details, see CreateGeofenceCollection in *AWS SDK for Kotlin API reference*.

`CreateKey`

The following code example shows how to use `CreateKey`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
```

```
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}
```

- For API details, see CreateKey in *AWS SDK for Kotlin API reference*.

`CreateMap`

The following code example shows how to use `CreateMap`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
```

```
    }

    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}
```

- For API details, see CreateMap in *AWS SDK for Kotlin API reference*.

`CreateRouteCalculator`

The following code example shows how to use `CreateRouteCalculator`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }
```

```
    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}
```

- For API details, see CreateRouteCalculator in *AWS SDK for Kotlin API reference*.

`CreateTracker`

The following code example shows how to use `CreateTracker`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the Amazon
 Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}
```

- For API details, see CreateTracker in *AWS SDK for Kotlin API reference*.

`DeleteGeofenceCollection`

The following code example shows how to use `DeleteGeofenceCollection`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence collection
 has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}
```

- For API details, see DeleteGeofenceCollection in *AWS SDK for Kotlin API reference*.

`DeleteKey`

The following code example shows how to use `DeleteKey`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}
```

- For API details, see DeleteKey in *AWS SDK for Kotlin API reference*.

`DeleteMap`

The following code example shows how to use `DeleteMap`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
```

```
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}
```

- For API details, see DeleteMap in *AWS SDK for Kotlin API reference*.

`DeleteRouteCalculator`

The following code example shows how to use `DeleteRouteCalculator`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
```

```
    }
}
```

- For API details, see DeleteRouteCalculator in *AWS SDK for Kotlin API reference*.

`DeleteTracker`

The following code example shows how to use `DeleteTracker`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}
```

- For API details, see DeleteTracker in *AWS SDK for Kotlin API reference*.

`GetDevicePosition`

The following code example shows how to use `GetDevicePosition`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}
```

- For API details, see GetDevicePosition in *AWS SDK for Kotlin API reference*.

`PutGeofence`

The following code example shows how to use `PutGeofence`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}
```

- For API details, see PutGeofence in *AWS SDK for Kotlin API reference*.

MediaConvert examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with MediaConvert.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Actions

Actions

`CreateJob`

The following code example shows how to use `CreateJob`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createMediaJob(
    mcClient: MediaConvertClient,
    mcRoleARN: String,
    fileInput1: String,
): String? {
    // Step 1: Describe endpoints to get the MediaConvert endpoint URL
    val describeResponse = mcClient.describeEndpoints(
        DescribeEndpointsRequest {
            maxResults = 1
        },
    )

    val endpointUrl = describeResponse.endpoints?.firstOrNull()?.url
        ?: error("No MediaConvert endpoint found")

    // Step 2: Create MediaConvert client with resolved endpoint
    val mediaConvert = MediaConvertClient.fromEnvironment {
        region = "us-west-2"
        endpointProvider = MediaConvertEndpointProvider {
            Endpoint(endpointUrl)
        }
```

```
    }

    // Output destination folder in S3 - put in 'output/' folder beside input
    val outputDestination = fileInput1.substringBeforeLast('/') + "/output/"

    // Step 3: Create the job request with minimal valid video codec settings
    val jobRequest = CreateJobRequest {
        role = mcRoleARN
        settings = JobSettings {
            inputs = listOf(
                Input {
                    fileInput = fileInput1
                },
            )
            outputGroups = listOf(
                OutputGroup {
                    outputGroupSettings = OutputGroupSettings {
                        type = OutputGroupType.FileGroupSettings
                        fileGroupSettings = FileGroupSettings {
                            destination = outputDestination
                        }
                    }
                    outputs = listOf(
                        Output {
                            containerSettings = ContainerSettings {
                                container = ContainerType.Mp4
                            }
                            videoDescription = VideoDescription {
                                width = 1280
                                height = 720
                                codecSettings = VideoCodecSettings {
                                    codec = VideoCodec.H264
                                    h264Settings = H264Settings {
                                        rateControlMode = H264RateControlMode.Qvbr
                                        qvbrSettings = H264QvbrSettings {
                                            qvbrQualityLevel = 7
                                        }
                                        maxBitrate = 5_000_000
                                        codecLevel = H264CodecLevel.Auto
                                        codecProfile = H264CodecProfile.Main
                                        framerateControl =
H264FramerateControl.InitializeFromSource
                                    }
                                }
```

```
                            }
                        },
                    )
                },
            )
        }
    }

    // Step 4: Call MediaConvert to create the job
    val response = mediaConvert.createJob(jobRequest)

    // Return the job ID or null if not found
    return response.job?.id
}
```

- For API details, see CreateJob in *AWS SDK for Kotlin API reference*.

`GetJob`

The following code example shows how to use `GetJob`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getSpecificJob(mcClient: MediaConvertClient, jobId: String) {
    // 1. Discover the correct endpoint
    val res = mcClient.describeEndpoints(DescribeEndpointsRequest { maxResults =
 1 })
    var endpointUrl = res.endpoints?.firstOrNull()?.url
        ?: error(" No MediaConvert endpoint found")

    // 2. Create a new client using the endpoint
    val clientWithEndpoint = MediaConvertClient {
        region = "us-west-2"
        endpointUrl = endpointUrl
    }
```

```
    // 3. Get the job details
    val jobResponse = clientWithEndpoint.getJob(GetJobRequest { id = jobId })
    val job = jobResponse.job

    println("Job status: ${job?.status}")
    println("Job ARN: ${job?.arn}")
    println("Output group count: ${job?.settings?.outputGroups?.size}")
}
```

- For API details, see GetJob in *AWS SDK for Kotlin API reference*.

`ListJobs`

The following code example shows how to use `ListJobs`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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}")
        }
    }
}
```

- For API details, see ListJobs in *AWS SDK for Kotlin API reference*.

Amazon Pinpoint examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Pinpoint.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Actions

Actions

`CreateApp`

The following code example shows how to use `CreateApp`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequestOb =
        CreateApplicationRequest {
            name = applicationName
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequestOb
                },
            )
        return result.applicationResponse?.id
    }
}
```

- For API details, see CreateApp in *AWS SDK for Kotlin API reference*.

`CreateCampaign`

The following code example shows how to use `CreateCampaign`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
            pinpoint.createCampaign(
                CreateCampaignRequest {
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- For API details, see CreateCampaign in *AWS SDK for Kotlin API reference*.

`CreateSegment`

The following code example shows how to use `CreateSegment`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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
    }
}
```

- For API details, see CreateSegment in *AWS SDK for Kotlin API reference*.

`DeleteApp`

The following code example shows how to use `DeleteApp`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deletePinApp(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
```

```
            pinpoint.deleteApp(
                DeleteAppRequest {
                    applicationId = appId
                },
            )
        val appName = result.applicationResponse?.name
        println("Application $appName has been deleted.")
    }
}
```

- For API details, see DeleteApp in *AWS SDK for Kotlin API reference*.

`DeleteEndpoint`

The following code example shows how to use `DeleteEndpoint`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deletePinEncpoint(
    appIdVal: String?,
    endpointIdVal: String?,
) {
    val deleteEndpointRequest =
        DeleteEndpointRequest {
            applicationId = appIdVal
            endpointId = endpointIdVal
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is  $id")
    }
}
```

- For API details, see DeleteEndpoint in *AWS SDK for Kotlin API reference*.

`GetEndpoint`

The following code example shows how to use `GetEndpoint`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun lookupPinpointEndpoint(
    appId: String?,
    endpoint: String?,
) {
    PinpointClient.fromEnvironment { 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)
    }
}
```

- For API details, see GetEndpoint in *AWS SDK for Kotlin API reference*.

`GetSegments`

The following code example shows how to use `GetSegments`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listSegs(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val response =
            pinpoint.getSegments(
                GetSegmentsRequest {
                    applicationId = appId
                },
            )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
    }
}
```

- For API details, see GetSegments in *AWS SDK for Kotlin API reference*.

`SendMessages`

The following code example shows how to use `SendMessages`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
Before running this 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.fromEnvironment { region = "us-east-1" }.use { pinpointemail
 ->
        pinpointemail.sendEmail(sendEmailRequest)
        println("Message Sent")
    }
}
```

- For API details, see SendMessages in *AWS SDK for Kotlin API reference*.

Amazon RDS examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon RDS.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Basics
- Actions
- Scenarios

Basics

Learn the basics

The following code example shows how to:

- Create a custom DB parameter group and set parameter values.
- Create a DB instance that's configured to use the parameter group. The DB instance also contains a database.
- Take a snapshot of the instance.
- Delete the instance and parameter group.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
Before running this code 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.t3.micro"
            engineVersion = "8.0.35"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-west-2" }.use
 { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Actions

`CreateDBInstance`

The following code example shows how to use `CreateDBInstance`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNamedbVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro" // Use a supported instance class
            engineVersion = "8.0.39" // Use a supported engine version
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { 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.fromEnvironment { 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!")
    }
}
```

- For API details, see CreateDBInstance in *AWS SDK for Kotlin API reference*.

`DeleteDBInstance`

The following code example shows how to use `DeleteDBInstance`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- For API details, see DeleteDBInstance in *AWS SDK for Kotlin API reference*.

`DescribeAccountAttributes`

The following code example shows how to use `DescribeAccountAttributes`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getAccountAttributes() {
```

```
    RdsClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DescribeAccountAttributes in *AWS SDK for Kotlin API reference*.

`DescribeDBInstances`

The following code example shows how to use `DescribeDBInstances`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeInstances() {
    RdsClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DescribeDBInstances in *AWS SDK for Kotlin API reference*.

`ModifyDBInstance`

The following code example shows how to use `ModifyDBInstance`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- For API details, see ModifyDBInstance in *AWS SDK for Kotlin API reference*.

Scenarios

Create an Aurora Serverless work item tracker

The following code example shows how to create a web application that tracks work items in an Amazon Aurora Serverless database and uses Amazon Simple Email Service (Amazon SES) to send reports.

SDK for Kotlin

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on GitHub.

Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Amazon RDS Data Service examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon RDS Data Service.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Scenarios

Scenarios

Create an Aurora Serverless work item tracker

The following code example shows how to create a web application that tracks work items in an Amazon Aurora Serverless database and uses Amazon Simple Email Service (Amazon SES) to send reports.

SDK for Kotlin

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on GitHub.

Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Amazon Redshift examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Redshift.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Actions
- Scenarios

Actions

`CreateCluster`

The following code example shows how to use `CreateCluster`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the cluster.

```
suspend fun createCluster(
    clusterId: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val clusterRequest =
        CreateClusterRequest {
            clusterIdentifier = clusterId
            availabilityZone = "us-east-1a"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
            nodeType = "ra3.4xlarge"
            publiclyAccessible = true
            numberOfNodes = 2
        }

    RedshiftClient.fromEnvironment { region = "us-east-1" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}
```

- For API details, see CreateCluster in *AWS SDK for Kotlin API reference*.

`DeleteCluster`

The following code example shows how to use `DeleteCluster`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete the cluster.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
        DeleteClusterRequest {
            clusterIdentifier = clusterId
            skipFinalClusterSnapshot = true
        }

    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- For API details, see DeleteCluster in *AWS SDK for Kotlin API reference*.

`DescribeClusters`

The following code example shows how to use `DescribeClusters`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Describe the cluster.

```
suspend fun describeRedshiftClusters() {
```

```
    RedshiftClient.fromEnvironment { 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}")
            }
        }
    }
}
```

- For API details, see DescribeClusters in *AWS SDK for Kotlin API reference*.

`ModifyCluster`

The following code example shows how to use `ModifyCluster`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Modify a cluster.

```
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",
        )
    }
}
```

- For API details, see ModifyCluster in *AWS SDK for Kotlin API reference*.

Scenarios

Create a web application to track Amazon Redshift data

The following code example shows how to create a web application that tracks and reports on work items using an Amazon Redshift database.

SDK for Kotlin

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on GitHub.

Services used in this example

- Amazon Redshift
- Amazon SES

Amazon Rekognition examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Rekognition.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Actions
- Scenarios

Actions

`CompareFaces`

The following code example shows how to use `CompareFaces`.

For more information, see Comparing faces in images.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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}")
    }
}
```

- For API details, see CompareFaces in *AWS SDK for Kotlin API reference*.

`CreateCollection`

The following code example shows how to use `CreateCollection`.

For more information, see Creating a collection.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- For API details, see CreateCollection in *AWS SDK for Kotlin API reference*.

`DeleteCollection`

The following code example shows how to use `DeleteCollection`.

For more information, see Deleting a collection.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- For API details, see DeleteCollection in *AWS SDK for Kotlin API reference*.

`DeleteFaces`

The following code example shows how to use `DeleteFaces`.

For more information, see Deleting faces from a collection.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
```

```
        println("$faceIdVal was deleted from the collection")
    }
}
```

- For API details, see DeleteFaces in *AWS SDK for Kotlin API reference*.

`DescribeCollection`

The following code example shows how to use `DescribeCollection`.

For more information, see Describing a collection.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient.fromEnvironment { 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}")
    }
}
```

- For API details, see DescribeCollection in *AWS SDK for Kotlin API reference*.

`DetectFaces`

The following code example shows how to use `DetectFaces`.

For more information, see Detecting faces in an image.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DetectFaces in *AWS SDK for Kotlin API reference*.

`DetectLabels`

The following code example shows how to use `DetectLabels`.

For more information, see Detecting labels in an image.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- For API details, see DetectLabels in *AWS SDK for Kotlin API reference*.

`DetectModerationLabels`

The following code example shows how to use `DetectModerationLabels`.

For more information, see Detecting inappropriate images.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DetectModerationLabels in *AWS SDK for Kotlin API reference*.

`DetectText`

The following code example shows how to use `DetectText`.

For more information, see Detecting text in an image.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DetectText in *AWS SDK for Kotlin API reference*.

`IndexFaces`

The following code example shows how to use `IndexFaces`.

For more information, see Adding faces to a collection.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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")
            }
        }
    }
}
```

- For API details, see IndexFaces in *AWS SDK for Kotlin API reference*.

`ListCollections`

The following code example shows how to use `ListCollections`.

For more information, see Listing collections.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- For API details, see ListCollections in *AWS SDK for Kotlin API reference*.

`ListFaces`

The following code example shows how to use `ListFaces`.

For more information, see Listing faces in a collection.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see ListFaces in *AWS SDK for Kotlin API reference*.

`RecognizeCelebrities`

The following code example shows how to use `RecognizeCelebrities`.

For more information, see Recognizing celebrities in an image.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { 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.")
    }
}
```

- For API details, see RecognizeCelebrities in *AWS SDK for Kotlin API reference*.

Scenarios

Create a serverless application to manage photos

The following code example shows how to create a serverless application that lets users manage photos using labels.

SDK for Kotlin

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

Services used in this example

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

Detect information in videos

The following code example shows how to:

- Start Amazon Rekognition jobs to detect elements like people, objects, and text in videos.
- Check job status until jobs finish.
- Output the list of elements detected by each job.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Detect faces in a video stored in an Amazon S3 bucket.

```
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.fromEnvironment { 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.fromEnvironment { 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}")
        }
    }
}
```

Detect inappropriate or offensive content in a video stored in an Amazon S3 bucket.

```
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.fromEnvironment { 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)
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection

- StartTextDetection

Detect objects in images

The following code example shows how to build an app that uses Amazon Rekognition to detect objects by category in images.

SDK for Kotlin

Shows how to use Amazon Rekognition Kotlin API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

Route 53 domain registration examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Route 53 domain registration.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello Route 53 domain registration

The following code examples show how to get started using Route 53 domain registration.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 Before running this 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.fromEnvironment { 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}")
            }
    }
}
```

- For API details, see ListPrices in *AWS SDK for Kotlin API reference*.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- List current domains, and list operations in the past year.
- View billing for the past year, and view prices for domain types.
- Get domain suggestions.
- Check domain availability and transferability.
- Optionally, request a domain registration.
- Get an operation detail.
- Optionally, get a domain detail.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
Before running this 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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}")
            }
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Actions

`CheckDomainAvailability`

The following code example shows how to use `CheckDomainAvailability`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}
```

- For API details, see CheckDomainAvailability in *AWS SDK for Kotlin API reference*.

`CheckDomainTransferability`

The following code example shows how to use `CheckDomainTransferability`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
```

```
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}
```

- For API details, see CheckDomainTransferability in *AWS SDK for Kotlin API reference*.

`GetDomainDetail`

The following code example shows how to use `GetDomainDetail`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { 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}")
    }
}
```

- For API details, see GetDomainDetail in *AWS SDK for Kotlin API reference*.

`GetDomainSuggestions`

The following code example shows how to use `GetDomainSuggestions`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient.fromEnvironment { 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(" ")
        }
    }
}
```

- For API details, see GetDomainSuggestions in *AWS SDK for Kotlin API reference*.

`GetOperationDetail`

The following code example shows how to use `GetOperationDetail`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

- For API details, see GetOperationDetail in *AWS SDK for Kotlin API reference*.

`ListDomains`

The following code example shows how to use `ListDomains`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listDomains() {
    Route53DomainsClient.fromEnvironment { 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}")
            }
    }
}
```

- For API details, see ListDomains in *AWS SDK for Kotlin API reference*.

`ListOperations`

The following code example shows how to use `ListOperations`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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}")
            }
    }
}
```

- For API details, see ListOperations in *AWS SDK for Kotlin API reference*.

`ListPrices`

The following code example shows how to use `ListPrices`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { 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}")
            }
```

```
    }
}
```

- For API details, see ListPrices in *AWS SDK for Kotlin API reference*.

`RegisterDomain`

The following code example shows how to use `RegisterDomain`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}
```

- For API details, see RegisterDomain in *AWS SDK for Kotlin API reference*.

`ViewBilling`

The following code example shows how to use `ViewBilling`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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}")
            }
    }
}
```

- For API details, see ViewBilling in *AWS SDK for Kotlin API reference*.

Amazon S3 examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon S3.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Basics

- Actions
- Scenarios

Basics

Learn the basics

The following code example shows how to:

- Create a bucket and upload a file to it.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the bucket objects and the bucket.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2

- PutObject

Actions

`CopyObject`

The following code example shows how to use `CopyObject`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- For API details, see CopyObject in *AWS SDK for Kotlin API reference*.

`CreateBucket`

The following code example shows how to use `CreateBucket`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- For API details, see CreateBucket in *AWS SDK for Kotlin API reference*.

`CreateMultiRegionAccessPoint`

The following code example shows how to use `CreateMultiRegionAccessPoint`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Configure the S3 control client to send request to the us-west-2 Region.

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

Create the Multi-Region Access Point.

```
    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"
    }
```

Wait for the Multi-Region Access Point to become available.

```
        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)
```

```
            }
        }
```

- For more information, see AWS SDK for Kotlin developer guide.
- For API details, see CreateMultiRegionAccessPoint in *AWS SDK for Kotlin API reference*.

`DeleteBucketPolicy`

The following code example shows how to use `DeleteBucketPolicy`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- For API details, see DeleteBucketPolicy in *AWS SDK for Kotlin API reference*.

`DeleteObjects`

The following code example shows how to use `DeleteObjects`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- For API details, see DeleteObjects in *AWS SDK for Kotlin API reference*.

`GetBucketPolicy`

The following code example shows how to use `GetBucketPolicy`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- For API details, see GetBucketPolicy in *AWS SDK for Kotlin API reference*.

`GetObject`

The following code example shows how to use `GetObject`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
```

```
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- For API details, see GetObject in *AWS SDK for Kotlin API reference*.

`GetObjectAcl`

The following code example shows how to use `GetObjectAcl`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }
```

```
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- For API details, see GetObjectAcl in *AWS SDK for Kotlin API reference*.

`ListObjectsV2`

The following code example shows how to use `ListObjectsV2`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { 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
```

- For API details, see ListObjectsV2 in *AWS SDK for Kotlin API reference*.

`PutBucketAcl`

The following code example shows how to use `PutBucketAcl`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- For API details, see PutBucketAcl in *AWS SDK for Kotlin API reference*.

`PutObject`

The following code example shows how to use `PutObject`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- For API details, see PutObject in *AWS SDK for Kotlin API reference*.

Scenarios

Create a presigned URL

The following code example shows how to create a presigned URL for Amazon S3 and upload an object.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a `GetObject` presigned request and use the URL to download an object.

```
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
}
```

Create a `GetObject` presigned request with advanced options.

```
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
}
```

Create a `PutObject` presigned request and use it to upload an object.

```
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)
}
```

- For more information, see AWS SDK for Kotlin developer guide.

Create a serverless application to manage photos

The following code example shows how to create a serverless application that lets users manage photos using labels.

SDK for Kotlin

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

Services used in this example

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

Detect objects in images

The following code example shows how to build an app that uses Amazon Rekognition to detect objects by category in images.

SDK for Kotlin

Shows how to use Amazon Rekognition Kotlin API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

Get an object from a Multi-Region Access Point

The following code example shows how to get an object from a Multi-Region Access Point.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Configure the S3 client to use the Asymmetric Sigv4 (Sigv4a) signing algorithm.

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric SigV4 (SigV4a) signing
algorithm.
            val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4aScheme)
            }
            return s3
        }
```

Use the Multi-Region Access Point ARN instead of a bucket name to retrieve the object.

```
    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
```

```
    }
```

- For more information, see AWS SDK for Kotlin developer guide.
- For API details, see GetObject in *AWS SDK for Kotlin API reference*.

SageMaker AI examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with SageMaker AI.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello SageMaker AI

The following code examples show how to get started using SageMaker AI.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listBooks() {
    SageMakerClient.fromEnvironment { region = "us-west-2" }.use { sageMakerClient -
>
        val response =
 sageMakerClient.listNotebookInstances(ListNotebookInstancesRequest {})
        response.notebookInstances?.forEach { item ->
```

```
            println("The notebook name is: ${item.notebookInstanceName}")
        }
    }
}
```

- For API details, see ListNotebookInstances in *AWS SDK for Kotlin API reference*.

Topics

- Actions
- Scenarios

Actions

`CreatePipeline`

The following code example shows how to use `CreatePipeline`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Create a 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)
        }
    }
}
```

- For API details, see CreatePipeline in *AWS SDK for Kotlin API reference*.

`DeletePipeline`

The following code example shows how to use `DeletePipeline`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Delete 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")
    }
}
```

- For API details, see DeletePipeline in *AWS SDK for Kotlin API reference*.

`DescribePipelineExecution`

The following code example shows how to use `DescribePipelineExecution`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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")
}
```

- For API details, see DescribePipelineExecution in *AWS SDK for Kotlin API reference*.

`StartPipelineExecution`

The following code example shows how to use `StartPipelineExecution`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// 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
    }
}
```

- For API details, see StartPipelineExecution in *AWS SDK for Kotlin API reference*.

Scenarios

Get started with geospatial jobs and pipelines

The following code example shows how to:

- Set up resources for a pipeline.
- Set up a pipeline that executes a geospatial job.
- Start a pipeline execution.
- Monitor the status of the execution.
- View the output of the pipeline.
- Clean up resources.

For more information, see Create and run SageMaker pipelines using AWS SDKs on Community.aws.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Secrets Manager examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Secrets Manager.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Actions

Actions

`GetSecretValue`

The following code example shows how to use `GetSecretValue`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-east-1" }.use
 { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
```

```
    }
}
```

- For API details, see GetSecretValue in *AWS SDK for Kotlin API reference*.

Amazon SES examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon SES.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Scenarios

Scenarios

Create a web application to track DynamoDB data

The following code example shows how to create a web application that tracks work items in an Amazon DynamoDB table and uses Amazon Simple Email Service (Amazon SES) to send reports.

SDK for Kotlin

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- DynamoDB
- Amazon SES

Create a web application to track Amazon Redshift data

The following code example shows how to create a web application that tracks and reports on work items using an Amazon Redshift database.

SDK for Kotlin

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on GitHub.

Services used in this example

- Amazon Redshift
- Amazon SES

Create an Aurora Serverless work item tracker

The following code example shows how to create a web application that tracks work items in an Amazon Aurora Serverless database and uses Amazon Simple Email Service (Amazon SES) to send reports.

SDK for Kotlin

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on GitHub.

Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Detect objects in images

The following code example shows how to build an app that uses Amazon Rekognition to detect objects by category in images.

SDK for Kotlin

Shows how to use Amazon Rekognition Kotlin API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

Amazon SNS examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon SNS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon SNS

The following code examples show how to get started using Amazon SNS.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import 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.fromEnvironment { 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}")
            }
    }
}
```

- For API details, see ListTopics in *AWS SDK for Kotlin API reference*.

Topics

- Actions

- Scenarios

Actions

`CreateTopic`

The following code example shows how to use `CreateTopic`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- For API details, see CreateTopic in *AWS SDK for Kotlin API reference*.

`DeleteTopic`

The following code example shows how to use `DeleteTopic`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- For API details, see DeleteTopic in *AWS SDK for Kotlin API reference*.

`GetTopicAttributes`

The following code example shows how to use `GetTopicAttributes`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
```

```
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- For API details, see GetTopicAttributes in *AWS SDK for Kotlin API reference*.

`ListSubscriptions`

The following code example shows how to use `ListSubscriptions`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listSNSSubscriptions() {
    SnsClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see ListSubscriptions in *AWS SDK for Kotlin API reference*.

`ListTopics`

The following code example shows how to use `ListTopics`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listSNSTopics() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- For API details, see ListTopics in *AWS SDK for Kotlin API reference*.

`Publish`

The following code example shows how to use `Publish`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
```

```
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- For API details, see Publish in *AWS SDK for Kotlin API reference*.

`SetTopicAttributes`

The following code example shows how to use `SetTopicAttributes`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- For API details, see SetTopicAttributes in *AWS SDK for Kotlin API reference*.

`Subscribe`

The following code example shows how to use `Subscribe`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Subscribe an email address to a topic.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Subscribe a Lambda function to a topic.

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
```

```
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- For API details, see Subscribe in *AWS SDK for Kotlin API reference*.

`TagResource`

The following code example shows how to use `TagResource`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- For API details, see TagResource in *AWS SDK for Kotlin API reference*.

`Unsubscribe`

The following code example shows how to use `Unsubscribe`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- For API details, see Unsubscribe in *AWS SDK for Kotlin API reference*.

Scenarios

Building an Amazon SNS application

The following code example shows how to create an application that has subscription and publish functionality and translates messages.

SDK for Kotlin

Shows how to use the Amazon SNS Kotlin API to create an application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to create a web app, see the full example on GitHub.

For complete source code and instructions on how to create a native Android app, see the full example on GitHub.

Services used in this example

- Amazon SNS
- Amazon Translate

Create a serverless application to manage photos

The following code example shows how to create a serverless application that lets users manage photos using labels.

SDK for Kotlin

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

Services used in this example

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

Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- For API details, see Publish in *AWS SDK for Kotlin API reference*.

Publish messages to queues

The following code example shows how to:

- Create topic (FIFO or non-FIFO).
- Subscribe several queues to the topic with an option to apply a filter.
- Publish messages to the topic.
- Poll the queues for messages received.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package com.example.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 scenario, 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
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Amazon SQS examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon SQS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon SQS

The following code examples show how to get started using Amazon SQS.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package com.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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
```

```
        sqsClient
            .listQueuesPaginated { }
            .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
            .collect { queue ->
                println("The Queue URL is $queue")
            }
    }
}
```

- For API details, see ListQueues in *AWS SDK for Kotlin API reference*.

Topics

- Actions
- Scenarios

Actions

`CreateQueue`

The following code example shows how to use `CreateQueue`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }

    SqsClient.fromEnvironment { 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()
    }
}
```

- For API details, see CreateQueue in *AWS SDK for Kotlin API reference*.

`DeleteMessage`

The following code example shows how to use `DeleteMessage`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- For API details, see DeleteMessage in *AWS SDK for Kotlin API reference*.

`DeleteQueue`

The following code example shows how to use `DeleteQueue`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- For API details, see DeleteQueue in *AWS SDK for Kotlin API reference*.

`ListQueues`

The following code example shows how to use `ListQueues`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest =
        ListQueuesRequest {
            queueNamePrefix = prefix
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
```

```
}
```

- For API details, see ListQueues in *AWS SDK for Kotlin API reference*.

`ReceiveMessage`

The following code example shows how to use `ReceiveMessage`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- For API details, see ReceiveMessage in *AWS SDK for Kotlin API reference*.

`SendMessage`

The following code example shows how to use `SendMessage`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun sendMessages(
    queueUrlVal: String,
    message: String,
) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest =
        SendMessageRequest {
            queueUrl = queueUrlVal
            messageBody = message
            delaySeconds = 10
        }

    SqsClient.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- For API details, see SendMessage in *AWS SDK for Kotlin API reference*.

Scenarios

Create a messaging application

The following code example shows how to create a messaging application by using Amazon SQS.

SDK for Kotlin

Shows how to use the Amazon SQS API to develop a Spring REST API that sends and retrieves messages.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Comprehend
- Amazon SQS

Publish messages to queues

The following code example shows how to:

- Create topic (FIFO or non-FIFO).
- Subscribe several queues to the topic with an option to apply a filter.
- Publish messages to the topic.

- Poll the queues for messages received.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package com.example.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 scenario, 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
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Step Functions examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Step Functions.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello Step Functions

The following code examples show how to get started using Step Functions.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import 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.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see ListStateMachines in *AWS SDK for Kotlin API reference*.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- Create an activity.
- Create a state machine from an Amazon States Language definition that contains the previously created activity as a step.
- Run the state machine and respond to the activity with user input.
- Get the final status and output after the run completes, then clean up resources.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import 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.
        jsonFile - The location of the chat_sfn_state_machine.json file. You can
 located it in resources/sample_files.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(0)
    }

    val roleName = args[0]
    val activityName = args[1]
    val stateMachineName = args[2]
    val jsonFile = args[3]
    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(jsonFile)

    // 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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 $status")
            }
        }
    }
    println("The Status is $status")
}

suspend fun sendTaskSuccess(
    token: String?,
```

```
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Actions

`CreateActivity`

The following code example shows how to use `CreateActivity`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- For API details, see CreateActivity in *AWS SDK for Kotlin API reference*.

`CreateStateMachine`

The following code example shows how to use `CreateStateMachine`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}
```

- For API details, see CreateStateMachine in *AWS SDK for Kotlin API reference*.

`DeleteActivity`

The following code example shows how to use `DeleteActivity`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
```

```
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}
```

- For API details, see DeleteActivity in *AWS SDK for Kotlin API reference*.

`DeleteStateMachine`

The following code example shows how to use `DeleteStateMachine`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- For API details, see DeleteStateMachine in *AWS SDK for Kotlin API reference*.

`DescribeExecution`

The following code example shows how to use `DescribeExecution`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient.fromEnvironment { 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 $status")
            }
        }
    }
    println("The Status is $status")
}
```

- For API details, see DescribeExecution in *AWS SDK for Kotlin API reference*.

`DescribeStateMachine`

The following code example shows how to use `DescribeStateMachine`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient.fromEnvironment { 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}")
    }
}
```

- For API details, see DescribeStateMachine in *AWS SDK for Kotlin API reference*.

`GetActivityTask`

The following code example shows how to use `GetActivityTask`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
```

```
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}
```

- For API details, see GetActivityTask in *AWS SDK for Kotlin API reference*.

`ListActivities`

The following code example shows how to use `ListActivities`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listAllActivites() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see ListActivities in *AWS SDK for Kotlin API reference*.

`ListExecutions`

The following code example shows how to use `ListExecutions`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getExeHistory(exeARN: String?) {
    val historyRequest =
        GetExecutionHistoryRequest {
            executionArn = exeARN
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- For API details, see ListExecutions in *AWS SDK for Kotlin API reference*.

`ListStateMachines`

The following code example shows how to use `ListStateMachines`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import 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.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see ListStateMachines in *AWS SDK for Kotlin API reference*.

`SendTaskSuccess`

The following code example shows how to use `SendTaskSuccess`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- For API details, see SendTaskSuccess in *AWS SDK for Kotlin API reference*.

`StartExecution`

The following code example shows how to use `StartExecution`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}
```

- For API details, see StartExecution in *AWS SDK for Kotlin API reference*.

Support examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Support.

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Get started

Hello Support

The following code examples show how to get started using Support.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
Before running this 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.fromEnvironment { 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++
            }
        }
    }
}
```

- For API details, see DescribeServices in *AWS SDK for Kotlin API reference*.

Topics

- Basics
- Actions

Basics

Learn the basics

The following code example shows how to:

- Get and display available services and severity levels for cases.
- Create a support case using a selected service, category, and severity level.
- Get and display a list of open cases for the current day.
- Add an attachment set and a communication to the new case.
- Describe the new attachment and communication for the case.
- Resolve the case.
- Get and display a list of resolved cases for the current day.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
Before running this 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Actions

`AddAttachmentsToSet`

The following code example shows how to use `AddAttachmentsToSet`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

- For API details, see AddAttachmentsToSet in *AWS SDK for Kotlin API reference*.

`AddCommunicationToCase`

The following code example shows how to use `AddCommunicationToCase`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient.fromEnvironment { 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")
        }
    }
}
```

- For API details, see AddCommunicationToCase in *AWS SDK for Kotlin API reference*.

`CreateCase`

The following code example shows how to use `CreateCase`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- For API details, see CreateCase in *AWS SDK for Kotlin API reference*.

`DescribeAttachment`

The following code example shows how to use `DescribeAttachment`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}
```

- For API details, see DescribeAttachment in *AWS SDK for Kotlin API reference*.

`DescribeCases`

The following code example shows how to use `DescribeCases`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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.fromEnvironment { 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}")
        }
    }
}
```

- For API details, see DescribeCases in *AWS SDK for Kotlin API reference*.

`DescribeCommunications`

The following code example shows how to use `DescribeCommunications`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient.fromEnvironment { 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 ""
}
```

- For API details, see DescribeCommunications in *AWS SDK for Kotlin API reference*.

`DescribeServices`

The following code example shows how to use `DescribeServices`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { 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
}
```

- For API details, see DescribeServices in *AWS SDK for Kotlin API reference*.

`DescribeSeverityLevels`

The following code example shows how to use `DescribeSeverityLevels`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { 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
    }
}
```

- For API details, see DescribeSeverityLevels in *AWS SDK for Kotlin API reference*.

`ResolveCase`

The following code example shows how to use `ResolveCase`.

SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- For API details, see ResolveCase in *AWS SDK for Kotlin API reference*.

Amazon Translate examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Translate.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

Each example includes a link to the complete source code, where you can find instructions on how to set up and run the code in context.

Topics

- Scenarios

Scenarios

Building an Amazon SNS application

The following code example shows how to create an application that has subscription and publish functionality and translates messages.

SDK for Kotlin

Shows how to use the Amazon SNS Kotlin API to create an application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to create a web app, see the full example on GitHub.

For complete source code and instructions on how to create a native Android app, see the full example on GitHub.

Services used in this example

- Amazon SNS
- Amazon Translate

Security for the AWS SDK for Kotlin

Cloud security at Amazon Web Services (AWS) is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations. Security is a shared responsibility between AWS and you. The Shared Responsibility Model describes this as Security of the Cloud and Security in the Cloud.

Security of the Cloud – AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud and providing you with services that you can use securely. Our security responsibility is the highest priority at AWS, and the effectiveness of our security is regularly tested and verified by third-party auditors as part of the AWS Compliance Programs.

Security in the Cloud – Your responsibility is determined by the AWS service you are using, and other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

Topics

- Data protection in AWS SDK for Kotlin
- AWS SDK for Kotlin support for TLS 1.2
- Identity and Access Management
- Compliance Validation for this AWS Product or Service
- Resilience for this AWS Product or Service
- Infrastructure Security for this AWS Product or Service

Data protection in AWS SDK for Kotlin

The AWS shared responsibility model applies to data protection in AWS SDK for Kotlin. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy

FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with SDK for Kotlin or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

AWS SDK for Kotlin support for TLS 1.2

The following information applies only to Java SSL implementation (the default SSL implementation in the AWS SDK for Kotlin targeting the JVM). If you're using a different SSL implementation, see your specific SSL implementation to learn how to enforce TLS versions.

TLS support in Java

TLS 1.2 is supported starting in Java 7.

How to check the TLS version

To check what TLS version is supported in your Java virtual machine (JVM), you can use the following code.

```
println(SSLContext.getDefault().supportedSSLParameters.protocols.joinToString(separator
 = ", "))
```

To see the SSL handshake in action and what version of TLS is used, you can use the system property **javax.net.debug**.

```
-Djavax.net.debug=ssl
```

Identity and Access Management

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How AWS services work with IAM
- Troubleshooting AWS identity and access

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS.

Service user – If you use AWS services to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS features to do your work, you might need additional permissions. Understanding how access is managed can help you

request the right permissions from your administrator. If you cannot access a feature in AWS, see Troubleshooting AWS identity and access or the user guide of the AWS service you are using.

Service administrator – If you're in charge of AWS resources at your company, you probably have full access to AWS. It's your job to determine which AWS features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS, see the user guide of the AWS service you are using.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS. To view example AWS identity-based policies that you can use in IAM, see the user guide of the AWS service you are using.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in

the *AWS IAM Identity Center User Guide* and AWS Multi-factor authentication in IAM in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier

to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Methods to assume a role in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.

- **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
- **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.
- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

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

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's

permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS services work with IAM

To get a high-level view of how AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

To learn how to use a specific AWS service with IAM, see the security section of the relevant service's User Guide.

Troubleshooting AWS identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS and IAM.

Topics

- I am not authorized to perform an action in AWS
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my AWS resources

I am not authorized to perform an action in AWS

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `awes:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *my-example-widget* resource by using the `awes:GetWidget` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to AWS.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my AWS resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS supports these features, see How AWS services work with IAM.

- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Compliance Validation for this AWS Product or Service

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security Compliance & Governance – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- HIPAA Eligible Services Reference – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.

- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

Resilience for this AWS Product or Service

The AWS global infrastructure is built around AWS Regions and Availability Zones.

AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking.

With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

Infrastructure Security for this AWS Product or Service

This AWS product or service uses managed services, and therefore is protected by the AWS global network security. For information about AWS security services and how AWS protects

infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access this AWS Product or Service through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

Document history

This topic describes important changes to the AWS SDK for Kotlin Developer Guide over the course of its history.

Change	Description	Date
the section called “Retries”	Retry content reorganized and details about retryable exceptions added.	September 10, 2025
Add information about how to cache credentials for a standalone credentials provider	Cache credentials with a standalone provider	June 17, 2025
the section called “Mocking”	Add information about mocking and using MockK with the SDK for Kotlin	April 30, 2025
Add information on how to resolve dependency conflicts with the SDK using Gradle	How do I resolve dependency conflicts?	April 15, 2025
Data integrity protection with checksums	Content updated with details about automatic checksum calculation.	January 16, 2025
Update default credentials provider chain content	The default credentials provider chain.	January 15, 2025
Update build file examples	Show build file elements as generated by Gradle version 8.11.1. Show use of BOM. Embed links to the latest version of artifacts.	December 18, 2024

Add DynamoDB Mapper (Developer Preview) topic	Map classes to DynamoDB items by using the DynamoDB Mapper (Developer Preview)	October 29, 2024
Update Amazon S3 bucket names	Amazon S3 checksums with AWS SDK for Kotlin	September 30, 2024
Add information the OkHttp4 Engine	Specify an HTTP engine type	September 26, 2024
Add information about AWS account-based endpoints for DynamoDB	Use AWS account-based endpoints	September 24, 2024
Add a Troubleshooting FAQs topic	Troubleshooting FAQs	September 18, 2024
Update OpenTelemetry configuration example and configuration of the default global telemetry provider	Observability	May 2, 2024
Provide more detail about the service client creation process	Create a service client	March 14, 2024
Add Multi-Region Access Point topic	Work with Amazon S3 Multi-Region Access Points by using the SDK for Kotlin	February 6, 2024
Add Gradle version catalog instructions	Gradle version catalog (tab)	December 19, 2023
General Availability release	AWS SDK for Kotlin Developer Guide	November 27, 2023
Update the client endpoints configuration section based on SDK updates	Client endpoints	August 25, 2023

Amazon S3 checksums	Section added on how to use flexible checksums with Amazon S3.	August 14, 2023
Add Observability topic	Observability	August 3, 2023
Add a topic that discusses retries	Retries	July 7, 2023
Update the HTTP client configuration section based on SDK updates	HTTP client configuration	June 6, 2023
Add HTTP presigning topic	Presigning requests	June 2, 2023
Add HTTP interceptors topic	HTTP interceptors	May 22, 2023
Support for automatic token refresh	Update instructions for single sign-on access.	May 18, 2023
Amazon S3 checksums	Add a section that describes how to use checksums with Amazon S3.	May 15, 2023
Override service client configuration	Add a section that describes how to override the configuration of a service client and describes how resources are affected.	May 8, 2023
Enforce a minimum TLS version	Add a section that describes the options to enforce a minimum TLS version.	May 3, 2023
Client endpoint configuration	Add a topic that discusses client endpoint configuration.	April 7, 2023

IAM best practices updates	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM.	March 22, 2023
Add an example Maven project file	Show an example of a Maven project file in addition to a project file in Gradle in the Set up topic.	December 2, 2022
Revise content of Developer Guide Preview	Content updated to reflect recent development work	October 4, 2022
AWS SDK for Kotlin Developer Preview release	AWS SDK for Kotlin	December 2, 2021
AWS SDK for Kotlin alpha release	Announcing new AWS SDK for Kotlin alpha release	August 30, 2021