AWS SDK for Kotlin

Developer Guide

AWS SDK for Kotlin: Developer Guide

Table of Contents

This is prerelease documentation for a service in preview release. It is subject to change.

Developer guide - AWS SDK for Kotlin

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

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

Get started with the SDK

If you're ready to get hands-on with the SDK, follow the Getting started (p. 2) tutorial.

To set up your development environment, see Setting up (p. 8).

To create and configure service clients for making requests to AWS services, see Configuring the SDK (p. 12). For information on making requests to Amazon S3, DynamoDB, Amazon EC2 and other AWS services, see Using the SDK (p. 16).

For use cases and examples of performing specific API operations, see Code examples (p. 21).

Maintenance and support for SDK major versions

For information about maintenance and support for SDK major versions and their underlying dependencies, see the following 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 AWS SDK for Kotlin developers:

- AWS developer blog
- Developer forums
- SDK source (GitHub)
- The AWS Code Sample Catalog
- @awsdevelopers (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, DynamoDB, and more.

This tutorial shows you how you can use Gradle to define dependencies for the AWS SDK for Kotlin and then create code that writes data to a DynamoDB table.

Follow these steps to complete this tutorial:

Step 1: Set up for this tutorial

Before you begin this tutorial, you need an active AWS account, an AWS Identity and Access Management (IAM) user with a programmatic access key and permissions to DynamoDB, and a Java development environment configured to use that access key as credentials for AWS services.

Follow these steps to set up for this tutorial:

Create an account

If you do not have an AWS account, visit the Amazon Web Services signup page and follow the on-screen prompts to create and activate a new account. For detailed instructions, see How do I create and activate a new AWS account?.

After you activate your new AWS account, follow the instructions in Creating your first IAM admin user and group in the IAM User Guide. Use this account instead of the root account when signing in to the AWS Management Console or making requests to AWS services programmatically. For more information, see Security best practices in IAM in the IAM User Guide.

Create an IAM user

To complete this tutorial, you need to use credentials for an IAM user that has read and write access to DynamoDB. To make requests to AWS services using the AWS SDK for Kotlin, create an access key to use as credentials.

1. Sign in to the IAM console
2. In the navigation pane on the left, choose **Users**. Then choose **Add user**.
3. Enter *TestSDK* as the **User name** and select the **Programmatic access** checkbox. Choose **Next: Permissions**.

4. Under **Set permissions**, select **Attach existing policies directly**.
5. In the list of policies, select the checkbox for the **AmazonDynamoDBFullAccess** policy. Choose **Next: Tags**.
6. Choose **Next: Review**. Then choose **Create user**.
7. On the *Success* screen, choose **Download .csv**.

   The downloaded file contains the Access Key ID and the Secret Access Key for this tutorial. Treat your Secret Access Key as a password; save in a trusted location and do not share it.

   Note
   You will **not** have another opportunity to download or copy the Secret Access Key.

Install Java and Gradle

Your development environment needs to have Java 8 or later and Gradle installed.

- For Java, use Oracle Java SE Development Kit, Amazon Corretto, Red Hat OpenJDK, or AdoptOpenJDK.
- For Gradle, go to https://gradle.org/install/.

Configure credentials

Configure your development environment with your Access Key ID and the Secret Access Key. The AWS SDK for Kotlin uses this access key as credentials when your application makes requests to AWS services.

1. In a text editor, create a new file with the following code:

```
[default]
aws_access_key_id = YOUR_AWS_ACCESS_KEY_ID
aws_secret_access_key = YOUR_AWS_SECRET_ACCESS_KEY
```

2. In the text file you just created, replace *YOUR_AWS_ACCESS_KEY* with your unique AWS access key ID, and replace *YOUR_AWS_SECRET_ACCESS_KEY* with your unique AWS secret access key.
3. Save the file without a file extension. Refer to the following table for the correct location and file name based on your operating system.

Operating system	File name
Windows	`C:\Users\USERNAME\.aws\credentials`
Linux, macOS, or Unix	`~/.aws/credentials`

Step 2: Create the project

To create the project for this tutorial, first use Gradle to create a Kotlin project. Then update the `gradle.build.kts` file with the required settings and dependencies for the AWS SDK for Kotlin.

To create a new project using Gradle:

1. Create a new directory called `get-started` in a location of your choice, for example, your Desktop or Home folder.
2. Open a terminal or command prompt window and navigate to the `get-started` directory you created.

3. Use the following command to create a new Gradle project configuration file (`build.gradle.kts`) and a basic Kotlin class.

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

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

- In the folder `get-started` that you created in the previous procedure, open the `build.gradle.kts` file.
- Replace its contents with the following code, and then save your changes.

```
import org.jetbrains.kotlin.gradle.tasks.KotlinCompile

plugins {
    kotlin("jvm") version "1.5.30"
    application
}

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

repositories {
    mavenCentral()
}

dependencies {
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core:1.5.0")
    implementation("aws.sdk.kotlin:dynamodb:0.9.4-beta")
    testImplementation(kotlin("test"))
}

tasks.withType<Test> {
    useJUnitPlatform()
}

tasks.withType<KotlinCompile>() {
    kotlinOptions.jvmTarget = "1.8"
}

application.mainClass.set("example.aws.getstarted.AppKt")
```

The `dependencies` section contains a dependency to the DynamoDB module of the AWS SDK for Kotlin. The Gradle compiler is configured to use Java 1.8 in the `tasks.withType<KotlinCompile>()` section.

Step 3: Write the code

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

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

```
package example.aws.getstarted
```

```
import kotlinx.coroutines.delay
import kotlinx.coroutines.runBlocking
import aws.smithy.kotlin.runtime.time.Instant
import aws.smithy.kotlin.runtime.time.epochMilliseconds

import aws.sdk.kotlin.runtime.UnknownServiceErrorException
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import aws.sdk.kotlin.services.dynamodb.model.AttributeDefinition
import aws.sdk.kotlin.services.dynamodb.model.AttributeValue
import aws.sdk.kotlin.services.dynamodb.model.CreateTableRequest
import aws.sdk.kotlin.services.dynamodb.model.DynamoDbException
import aws.sdk.kotlin.services.dynamodb.model.KeySchemaElement
import aws.sdk.kotlin.services.dynamodb.model.KeyType
import aws.sdk.kotlin.services.dynamodb.model.ScalarAttributeType
import aws.sdk.kotlin.services.dynamodb.model.TableStatus

fun main() = runBlocking {

    val dynamoDbClient = DynamoDbClient { region = "us-west-2" }
    val newTable = "TestSDK" + Instant.now().epochMilliseconds
    val key = "key"

    try {
        tutorialSetup(dynamoDbClient, newTable, key)

        println("Writing to table...")

        dynamoDbClient.putItem {
            tableName = newTable
            item = mapOf(
                key to AttributeValue.S("${key}_value")
            )
        }

        println("Completed writing to table.")
        println()

        cleanUp(dynamoDbClient, newTable)

    } catch (e: DynamoDbException) {
        println("ERROR (DynamoDbException): " + e.message)
    } catch (e: UnknownServiceErrorException) {
        println("ERROR (UnknownServiceErrorException): " + e.message)
    } finally {
        dynamoDbClient.close()
    }
    println("Exiting...")
}

private suspend fun tutorialSetup(dynamoDbClient: DynamoDbClient, newTable: String, key:
 String) {

    val createTableRequest = CreateTableRequest {
        tableName = newTable
        attributeDefinitions = listOf(
            AttributeDefinition {
                attributeName = key
                attributeType = ScalarAttributeType.S
            }
        )
        keySchema = listOf(
            KeySchemaElement {
                attributeName = key
                keyType = KeyType.Hash
            }
        )
```

```
        provisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }
    }
    println("Creating table: $newTable...")
    dynamoDbClient.createTable(createTableRequest)
    println("Waiting for table to be active...")
    var tableIsActive = dynamoDbClient.describeTable {
        tableName = newTable
    }.table?.tableStatus == TableStatus.Active
    do {
        if (!tableIsActive) {
            delay(500)
            tableIsActive = dynamoDbClient.describeTable {
                tableName = newTable
            }.table?.tableStatus == TableStatus.Active
        }
    } while(!tableIsActive)
    println("$newTable is ready.")
    println()
}

private suspend fun cleanUp(dynamoDbClient: DynamoDbClient, newTable: String) {
    println("Cleaning up...")
    println("Deleting table: $newTable...")
    dynamoDbClient.deleteTable {
        tableName = newTable
    }
    println("$newTable has been deleted.")
    println()
    println("Cleanup complete")
    println()
}
```

Step 4: Build and run the application

After the project is created and contains the example class, build and run the application. To view the table and its data, edit the code to remove the cleanup steps and then rebuild the project.

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

```
gradle run
```

When you run the application, it calls the CreateTable API operation to create a new table and then calls PutItem to insert a new record into the new DynamoDB table.

Afterward, it will also delete the table.

To see the results in the DynamoDB console

1. In `App.kt`, comment out the line `cleanUp(dynamoDbClient, newTable)` and save the file.
2. Rebuild the project and write to a new table by running `gradle run`.
3. Sign in to the DynamoDB console to view the new item in the newly-created table.

After you view the item in the table, clean up test resources by deleting the table.

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 with your new application, we recommend terminating or deleting any Amazon Web Services resources you created during this tutorial to avoid incurring any charges. You may also want to delete or archive the project folder (`get-started`) you created in Step 2.

To clean up the resources you created during this tutorial:

- In the DynamoDB console, delete any objects and any buckets created when you ran the application.
- In the IAM console, delete the *TestSDK* user.

  If you delete this user, also remove the contents of the `credentials` file you created during setup.

Next steps

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

- Configuring the AWS SDK for Kotlin (p. 12)
- Using the AWS SDK for Kotlin (p. 16)
- Security for the AWS SDK for Kotlin (p. 135)

Setting up the AWS 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, DynamoDB, and more.

This section provides information about how to set up your development environment and projects to use the AWS SDK for Kotlin.

Overview

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

- An active AWS account
- An AWS Identity and Access Management (IAM) user with:
  - A programmatic access key
  - Permissions to the AWS resources you'll access using your application
- A development environment with:
  - Your access key configured as credentials for AWS
  - Java 8 or later
  - A build automation tool

Create an account

If you do not have an AWS account, create and activate a new one.

To create a new account

- Visit the Amazon Web Services signup page.
- Follow the on-screen prompts to create and activate a new account.

  For more detailed instructions, see How do I create and activate a new AWS account?.
- After you activate your new AWS account, follow the instructions in Creating your first IAM admin user and group in the IAM User Guide.

  Use this account instead of the root account when signing in to the AWS Management Console or making requests to AWS services programmatically. For more information, see Security best practices in IAM in the IAM User Guide.

Create an IAM user and programmatic access key

To use the AWS SDK for Kotlin to access AWS services, you need an AWS account and AWS credentials. To increase the security of your AWS account, we recommend that you access AWS services with an IAM user account instead of using your AWS account credentials.

> **Note**
> For an overview of IAM users and why they are important for the security of your account, see AWS security credentials in the Amazon Web Services General Reference.

For instructions on creating an access key for an existing IAM user, see Programmatic access in the IAM User Guide.

To create an IAM user and programmatic acess key

1. Go to the IAM console (you may need to sign in to AWS first).
2. Click **Users** in the sidebar to view your IAM users.
3. If you don't have any IAM users set up, click **Create New Users** to create one.
4. Select the IAM user in the list that you'll use to access AWS.
5. Open the **Security Credentials** tab, and click **Create Access Key**.

   Note
   You can have a maximum of two active access keys for any given IAM user. If your IAM user has two access keys already, then you'll need to delete one of them before creating a new key.
6. On the resulting dialog box, click the **Download Credentials** button to download the credential file to your computer. Or you can click **Show User Security Credentials** to view the IAM user's access key ID and secret access key, which you can copy and paste.

   Important
   There is no way to obtain the secret access key once you close the dialog box. You can, however, delete its associated access key ID and create a new one.

Set default credentials and region

To make requests to AWS using the AWS SDK for Kotlin, you must use cryptographically-signed credentials issued by AWS. With AWS SDKs and Tools like the AWS SDK for Kotlin, you use a programmatic access key, consisting of an Access Key ID and and a Secret Access Key, as credentials. You should set your credentials as the default credentials for accessing AWS with your application.

If you already have an IAM account created, see Create an IAM user and programmatic access key (p. 8) for instructions on creating a programmatic access key.

You should also set a default AWS Region for accessing AWS with your application. Some operations require a Region to be set. For the best network performance, you can select a Region that is geographically near to you or your customers.

To set default credentials and region

The most common way to set the default credentials and AWS Region is to use the shared `config` and `credentials` files. You can also use environment variables to set the default credentials and Region.

Setting the default credentials

Select one of these options to set the default credentials:

- Set credentials in the AWS credentials profile file on your local system, located at:
  - `~/.aws/credentials` on Linux, macOS, or Unix
  - `C:\Users\USERNAME\.aws\credentials` on Windows

This file should contain lines in the following format:

```
[default]
aws_access_key_id = your_access_key_id
```

```
aws_secret_access_key = your_secret_access_key
```

Substitute your own AWS credentials values for the values *your_access_key_id* and *your_secret_access_key*.

- Set the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` environment variables.

To set these variables on Linux, macOS, or Unix, use **`export`**:

```
export AWS_ACCESS_KEY_ID=your_access_key_id
export AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

To set these variables on Windows, use **`set`**:

```
set AWS_ACCESS_KEY_ID=your_access_key_id
set AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

Setting the default region

Select one of these options to set the default Region:

- Set the AWS Region in the AWS config file on your local system, located at:
  - ~/.aws/config on Linux, macOS, or Unix
  - C:\Users\USERNAME\.aws\config on Windows

This file should contain lines in the following format:

```
[default]
region = your_aws_region
```

Substitute your desired AWS Region (for example, "us-east-1") for *your_aws_region*.

- Set the `AWS_REGION` environment variable.

On Linux, macOS, or Unix, use **`export`**:

```
export AWS_REGION=your_aws_region
```

On Windows, use **`set`**:

```
set AWS_REGION=your_aws_region
```

Where *your_aws_region* is the desired AWS Region name.

For additional information about setting credentials and Region, see The .aws/credentials and .aws/config files, AWS Region, and Using environment variables in the AWS SDKs and Tools Reference Guide.

Install Java and a build tool

Your development environment needs the following:

- Java 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/.

Next steps

Once you have your AWS account and development environment set up, create a Kotlin project using your preferred build tool. Then add dependencies for the AWS services that you access using your application.

Example `gradle.build.kts` file:

```
import org.jetbrains.kotlin.gradle.tasks.KotlinCompile

plugins {
    kotlin("jvm") version "1.5.30"
    application
}

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

repositories {
    mavenCentral()
}

dependencies {
    implementation("aws.sdk.kotlin:s3:0.9.4-beta")
    implementation("aws.sdk.kotlin:dynamodb:0.9.4-beta")
    implementation("aws.sdk.kotlin:iam:0.9.4-beta")
    implementation("aws.sdk.kotlin:cloudwatch:0.9.4-beta")
    implementation("aws.sdk.kotlin:cognito:0.9.4-beta")
    implementation("aws.sdk.kotlin:sns:0.9.4-beta")
    implementation("aws.sdk.kotlin:pinpoint:0.9.4-beta")
    testImplementation(kotlin("test"))
}

tasks.withType<Test> {
    useJUnitPlatform()
}

tasks.withType<KotlinCompile>() {
    kotlinOptions.jvmTarget = "1.8"
}
```

Configuring service clients for the AWS SDK for Kotlin

Creating service clients

To make a request to an AWS service, first instantiate an object to serve as a client for that service.

You must specify a region and a credentials provider as part of the service client instantiation. The SDK uses these values to route requests to the correct region and to sign requests with your credentials. You can specify these values programmatically in code, or have them automatically loaded from the environment.

The simplest way to create a new service client is by loading the region, credentials, and other configuration from the application environment. For example, this code snippet instantiates a `DynamoDbClient` object as a service client for DynamoDB.

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

The `fromEnvironment()` method creates a service client with the default configuration. It uses the default provider chain to load credentials and the AWS Region. If credentials or the region can't be determined from the environment that the application is running in, the call fails. Using the `fromEnvironment()` method can be particularly convenient with AWS Lambda functions or Amazon EC2 instances or in other situations where you want to decouple your code from your configuration.

Altenratively, you can supply configuration values on the service client. The following code snippet shows an example of one way to instantiate an `Ec2Client` object as a service client for Amazon EC2.

```
val ec2Client = Ec2Client {
    region = "us-west-2"
    credentialsProvider = ProfileCredentialsProvider(profileName = "myprofile")
}
```

When you use the configuration from your environment, you can override specific settings for that service client by providing a new value for each property you need to override. The following code snippet is an example of overriding the region.

```
val ec2Client = Ec2Client.fromEnvironment() {
    region = "us-east-1"
}
```

Configuring service clients

You can configure individual settings on the service client to override the defaults, such as the HTTP engine to use, the logging level, the retry configuration, or other settings. Refer to the following code snippet for a few examples.

```
val s3Client = S3Client.fromEnvironment() {
```

```
    sdkLogMode = SdkLogMode.LogRequest
    credentialsProvider = ProfileCredentialsProvider(profileName = "alternatecredentials")
    ...
}
```

To reuse a configuration across service clients, generate an `AwsClientConfig` object with the desired settings. Refer to the following code snippet for an example.

```
import aws.sdk.kotlin.runtime.client.AwsClientConfig
import aws.sdk.kotlin.runtime.config.fromEnvironment
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import aws.sdk.kotlin.services.s3.S3Client
// ...

val sharedConfig = AwsClientConfig.fromEnvironment()

val dynamoDbClient = DynamoDbClient(sharedConfig)
val s3Client = S3Client(sharedConfig)
```

You can override individual configuration settings on a shared configuration in the same way that you do for a client object-specific configuration. Refer to the following code snippet for an example.

```
val s3Client = S3Client(sharedConfig) {
    sdkLogMode = SdkLogMode.LogRequest
}
```

AWS Region selection

Regions enable you to access AWS services that physically reside 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.

Choosing a region

You can specify a region and the SDK will automatically choose an appropriate endpoint for you. Refer to the following code snippet for an example of how to set the region.

```
val s3 = S3Client { region = "us-west-2" }
```

> **Note**
> After you instantiate a service client, it's *immutable* and the region *cannot be changed*. If you are working with multiple AWS Regions for the same service, you create multiple clients—one per region.

See Regions and Endpoints for the current list of regions and their corresponding endpoints for all AWS services.

Automatically determine the Region from the environment

When running on Amazon EC2 or AWS Lambda, you might want to configure clients to use the same region that your code is running on. This decouples your code from the environment it's running in and makes it easier to deploy your application to multiple regions for lower latency or redundancy.

To use the default region provider chain to determine the region from the environment, use the client builder's `fromEnvironment` method.

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

If you don't explicitly set a region on a service client, the SDK uses the default region provider chain to determine the region to use.

Default region provider chain

The following is the region lookup process:

1. Any explicit region set on the builder itself takes precedence over anything else.
2. The `aws.region` JVM system property is checked. If it's set, that region is used to configure the client.
3. The `AWS_REGION` environment variable is checked. If it's set, that region is used to configure the client.

   Note
   This environment variable is set by the Lambda container.
4. The SDK checks the AWS shared configuration file (usually located at `~/.aws/config`). If the *region* property is present, the SDK uses it.
   - The `AWS_CONFIG_FILE` environment variable can be used to customize the location of the shared config file.
   - The `AWS_PROFILE` environment variable or the `aws.profile` system property 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 Amazon EC2 instance.
6. If the SDK still hasn't found a region by this point, client creation fails with an exception.

Note
When developing AWS applications, a common approach is to use the *shared configuration file* to set the region for local development, and rely on the default region provider chain to determine the region when running on AWS infrastructure. This greatly simplifies client creation and keeps your application portable. For more information, see The shared config and credentials files in the AWS SDKs and Tools Reference Guide.

Logging for the AWS SDK for Kotlin

The AWS SDK for Kotlin is instrumented with Slf4j, which is an abstraction layer that enables the use of any one of several logging systems at runtime.

Supported logging systems include the Java Logging Framework and Apache Log4j, among others. This topic shows you how to enable and configure the SDK's logging functionality.

To learn more about Log4j, see the Apache website.

To enable additional logging for requests and responses, set the `sdkLogMode` when constructing a service client, configure a compatible SLF4J logger, and set its logging level to DEBUG. The following code snippet demonstrates an example where requests are logged with the body and the responses are logged without the body.

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

```
// ...

val client = DynamoDbClient {
    // ...
    sdkLogMode = SdkLogMode.LogRequestWithBody + SdkLogMode.LogResponse
}
```

Important

Only use wire logging for debugging purposes. Since it logs the unencrypted data, it can log sensitive data; be certain to not to enable it in your production environments. Additionally, note that large requests or responses, such as uploading files to Amazon S3, may significantly reduce application performance.

Using the AWS SDK for Kotlin

After Setting up the SDK (p. 8) and Configuring the SDK (p. 12), you are ready to make requests to AWS services such as Amazon S3, DynamoDB, IAM, Amazon EC2, and more.

This chapter provides instructions and programming examples you can use with the AWS SDK for Kotlin for specific features and AWS services.

Making requests

After you configure a service client, you instantiate it and use its methods to make requests to that AWS service.

For example, this code snippet shows how to create a `RunInstancesRequest` object to create a new Amazon EC2 instance:

```
val ec2Client = Ec2Client.fromEnvironment()

val runInstancesRequest = RunInstancesRequest {
    imageId = amiId
    instanceType = InstanceType.T1Micro
    maxCount = 1
    minCount = 1
}

ec2Client.runInstances(runInstancesRequest)
```

Each non-streaming operation on service clients has DSL overload, which is a type-safe way you can use to define the inputs for operations. For example, the following code snippet is equivalent to the previous one.

```
val ec2Client = Ec2Client.fromEnvironment()
val runInstancesRequest = ec2Client.runInstances {
    imageId = amiId
    instanceType = InstanceType.T1Micro
    maxCount = 1
    minCount = 1
}
```

If you have one or more operations to perform sequentially with the same AWS service, you can use the `use` extension with a lambda to perform the operations. When you use this method, the service client automatically closes down correctly. See the following code snippet for an example of how to perform the same operation as above without first separately declaring a service client.

```
Ec2Client { region = "us-west-2" }.use { ec2Client ->
    ec2Client.runInstances {
        imageId = amiId
        instanceType = InstanceType.T1Micro
        maxCount = 1
        minCount = 1
    }
}
```

Making concurrent requests

The AWS SDK for Kotlin is natively asynchronous. Each operation is generated as a `suspend` function, which you call from a coroutine. For additional information about coroutines, refer to the official Kotlin documentation.

To make concurrent requests, use the async coroutine builder to process the response or the launch coroutine builder if you only need to verify that the operation succeeded. `async`returns a Deferred, a lightweight non-blocking future that represents a promise to provide a result later. `launch` returns a Job without any associated value. Refer to the following code snippet for an example of how to retrieve the size of two different keys in Amazon S3.

```
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-west-2" }
    val myBucket = "bucket-name"
    val key1 = "first-object-key"
    val key2 = "second-object-key"

    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

Since the SDK is natively asynchronous, you use the runBlocking coroutine builder to perform operations from blocking code. This will launch a new coroutine and block the current thread until the coroutine completes.

Streaming requests

To make requests containing binary data, you represent these as a `ByteStream` type. The SDK provides multiple convenience methods to supply a `ByteStream`, including the following:

- `ByteStream.fromFile(file: File)`

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

The following code snippet demonstrates basic usage of these methods.

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

For more information and a complete list of methods available in the SDK, refer to the `aws.smithy.kotlin.runtime.content` package in the AWS SDK for Kotlin API Reference.

Handling responses

When you make a request, you can use a response handler to process the response back from the AWS service.

For example, this code snippet shows an example of how to handle the response from Amazon EC2 by printing out the `instanceId` for the new instance from the request above.

```
val response = ec2Client.runInstances(runInstancesRequest)
println(response.instances?.get(0)?.instanceId)
```

Refer to the following code snippet for an example of how the complete request might look with `use`.

```
Ec2Client { region = "us-west-2" }.use { ec2Client ->
    val response = ec2Client.runInstances {
        imageId = amiId
        instanceType = InstanceType.T1Micro
        maxCount = 1
        minCount = 1
    }
    println(response.instances?.get(0)?.instanceId)
}
```

Streaming responses

Binary data are represented as a `ByteStream` type, an abstract read-only stream of bytes.

Responses that contain binary stream data, such as from a `getObject` request to Amazon S3, are handled with a lambda function instead of returning the response directly. This scopes access to the response to the function, simplifying lifetime management for both the caller and the SDK runtime. You must define what will be returned from within the lambda, since all associated resources are released as soon as the function returns.

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

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

// 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("SUCCESS: Wrote $contentSize bytes to $path")
```

The SDK provides various ways of consuming the `ByteStream` type. These include:

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

For more information and a complete list, refer to the `aws.smithy.kotlin.runtime.content` package in the AWS SDK for Kotlin API Reference.

Closing the client

When you no longer need the service client, close it.

```
ec2Client.close()
```

Note
Service clients extend the `Closable` interface. As a best practice - especially with short-lived code such as AWS Lambda functions - explicitly call the `close()` method.

Handling exceptions with the AWS SDK for Kotlin

After Setting up the SDK (p. 8) and Configuring the SDK (p. 12), you are ready to make requests to AWS services such as Amazon S3, DynamoDB, IAM, Amazon EC2, and more.

This section describes the types of exceptions that may be thrown by the SDK and how to handle them.

Handling exceptions

Exceptions thrown by the SDK include `AwsServiceException` and `ClientException`, and their sub-classes. An `AwsServiceException` is the most common form of exception the SDK will throw. These exceptions represent responses from the AWS service and indicate that your request was successfully sent to the AWS service but could not be processed. You can also handle a `ClientException`, which occurs when there's a problem on the client side (i.e., in your development or application environment), such a network connection failure.

This code snippet demonstrates one way to handle service exceptions while uploading a file to Amazon S3.

```
val s3Client = S3Client { region = "us-west-2" }

try {
    val putObjectRequest = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        this.body = Paths.get(pathToFile).asByteStream()
    }

    s3Client.putObject(putObjectRequest)

} catch (S3Exception e) {
    println("ERROR (S3Exception): " + e.sdkErrorMetadata.errorMessage())
    exitProcess(1)
}
```

Error metadata

Every service and client exception has a property `sdkErrorMetadata` on it. This is a typed property bag that can be used to retrieve additional details about the error. Several predefined extensions exist off the `AwsErrorMetadata` type directly. These include the following.

- The unique request id: `sdkErrorMetadata.requestId`
- The human readable message: `sdkErrorMetadata.errorMessage`

  Note
  This will usually match the `Exception.message`, but may contain more information, for example, if the exception was unknown to the service.
- The raw protocol response: `sdkErrorMetadata.protocolResponse`

The following code snippet is an example of accessing 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}")
}
```

SDK for Kotlin code examples

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

The examples are divided into the following categories:

Single-service actions

Code excerpts that show you how to call individual service functions.

Single-service scenarios

Code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Cross-service examples

Sample applications that work across multiple AWS services.

Examples

Single-service actions and scenarios using SDK for Kotlin

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

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Services

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.

Actions are code excerpts that show you how to call individual CloudWatch functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple CloudWatch functions.

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

Topics

Actions

Create an alarm that watches a metric

The following code example shows how to create an Amazon CloudWatch alarm that watches a metric.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun putMetricAlarm(alarmNameVal: String, instanceIdVal: String) {

    val dimensionOb = Dimension {
        name = "InstanceId"
        value = instanceIdVal
    }

    val request = PutMetricAlarmRequest {
        alarmName = alarmNameVal
        comparisonOperator = ComparisonOperator.GreaterThanThreshold
        evaluationPeriods = 1
        metricName = "CPUUtilization"
        namespace = "AWS/EC2"
        period = 60
        statistic = Statistic.fromValue("Average")
        threshold = 70.0
        actionsEnabled = false
        alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
 utilization exceeds 70%"
        unit = StandardUnit.fromValue("Seconds")
```

```
        dimensions = listOf(dimensionOb)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

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

Delete alarms

The following code example shows how to delete Amazon CloudWatch alarms.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteCWAlarm(alarmNameVal: String) {

    val request = DeleteAlarmsRequest {
        alarmNames = listOf(alarmNameVal)
    }

    CloudWatchClient { 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*.

Describe alarms for a metric

The following code example shows how to describe Amazon CloudWatch alarms for a metric.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun desCWAlarms() {

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(DescribeAlarmsRequest {})
        response.metricAlarms?.forEach { alarm ->
            println("Retrieved alarm ${alarm.alarmName}")
        }
```

```
    }
}
```

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

Disable alarm actions

The following code example shows how to disable Amazon CloudWatch alarm actions.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun disableActions(alarmName: String) {

    val request = DisableAlarmActionsRequest {
        alarmNames = listOf(alarmName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

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

Enable alarm actions

The following code example shows how to enable Amazon CloudWatch alarm actions.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun enableActions(alarm: String) {

    val request = EnableAlarmActionsRequest {
        alarmNames = listOf(alarm)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- For API details, see EnableAlarmActions 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 that show you how to call individual CloudWatch Logs functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple CloudWatch Logs functions.

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

Topics

Actions

Delete a subscription filter

The following code example shows how to delete an Amazon CloudWatch Logs subscription filter.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteSubFilter(filter: String?, logGroup: String?) {

    val request = DeleteSubscriptionFilterRequest {
        filterName = filter
        logGroupName = logGroup
    }

    CloudWatchLogsClient { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

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

Describe existing subscription filters

The following code example shows how to describe Amazon CloudWatch Logs existing subscription filters.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun describeFilters(logGroup: String) {

    val request = DescribeSubscriptionFiltersRequest {
        logGroupName = logGroup
        limit = 1
    }

    CloudWatchLogsClient { region = "us-west-2" }.use { cwlClient ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- For API details, see DescribeSubscriptionFilters 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 that show you how to call individual Amazon Cognito Identity Provider functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon Cognito Identity Provider functions.

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

Topics

Actions

Confirm a user

The following code example shows how to confirm an Amazon Cognito user.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal: String?)
 {

    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

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

Get a token to associate an MFA application with a user

The following code example shows how to get a token to associate an MFA application with an Amazon Cognito user.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {

    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}
```

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

Get information about a user

The following code example shows how to get information about an Amazon Cognito user.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {

    val userRequest = AdminGetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

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

List users

The following code example shows how to list Amazon Cognito users.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun listAllUsers(userPoolId: String) {

    val request = ListUsersRequest {
        this.userPoolId = userPoolId
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

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

Resend a confirmation code

The following code example shows how to resend an Amazon Cognito confirmation code.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {

    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

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

Respond to an authentication challenge

The following code example shows how to respond to an Amazon Cognito authentication challenge.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
// 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 respondToAuthChallengeRequest = RespondToAuthChallengeRequest {
        challengeName = ChallengeNameType.SoftwareTokenMfa
        clientId = clientIdVal
        challengeResponses = challengeResponsesOb
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val respondToAuthChallengeResult =
 identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

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

Sign up a user

The following code example shows how to sign up a user with Amazon Cognito.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal: String?,
 emailVal: String?) {

    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)

    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

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

Start authentication with administrator credentials

The following code example shows how to start authentication with Amazon Cognito and administrator credentials.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun initiateAuth(clientIdVal: String?, userNameVal: String, passwordVal:
 String): InitiateAuthResponse {

    val authParas = mutableMapOf <String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal
```

```
    val authRequest = InitiateAuthRequest {
        clientId = clientIdVal
        authParameters = authParas
        authFlow = AuthFlowType.UserPasswordAuth
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val response = identityProviderClient.initiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

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

Verify an MFA application with a user

The following code example shows how to verify an MFA application with an Amazon Cognito user.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {

    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val verifyResponse = identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- 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 a user with a user name, password, and email address.
- Confirm the user from a code sent in email.
- 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**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
/**
 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 = initiateAuth(clientId, userName, password)
    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 = initiateAuth(clientId, userName, password)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {

    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(userName: String, clientIdVal: String?,
 mfaCode: String, sessionVal: String?) {

    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val respondToAuthChallengeRequest = RespondToAuthChallengeRequest {
        challengeName = ChallengeNameType.SoftwareTokenMfa
```

```
        clientId = clientIdVal
        challengeResponses = challengeResponsesOb
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val respondToAuthChallengeResult =
 identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {

    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val verifyResponse = identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {

    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun initiateAuth(clientIdVal: String?, userNameVal: String, passwordVal:
 String): InitiateAuthResponse {

    val authParas = mutableMapOf <String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest = InitiateAuthRequest {
        clientId = clientIdVal
        authParameters = authParas
        authFlow = AuthFlowType.UserPasswordAuth
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val response = identityProviderClient.initiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

```
suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal: String?)
 {

    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}

suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {

    val userRequest = AdminGetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal: String?,
 emailVal: String?) {

    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)

    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
 ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- 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

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.

Actions are code excerpts that show you how to call individual DynamoDB functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple DynamoDB functions.

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

Topics

Actions

Create a table

The following code example shows how to create a DynamoDB table.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createNewTable(tableNameVal: String, key: String): String? {

    val attDef = AttributeDefinition {
        attributeName = key
        attributeType = ScalarAttributeType.S
    }

    val keySchemaVal = KeySchemaElement {
        attributeName = key
        keyType = KeyType.Hash
    }

    val provisionedVal = ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
```

```
    }

    val request = CreateTableRequest {
        attributeDefinitions = listOf(attDef)
        keySchema = listOf(keySchemaVal)
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        var tableArn: String
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists { // suspend call
            tableName = tableNameVal
        }
        tableArn = response.tableDescription!!.tableArn.toString()
        println("Table $tableArn is ready")
        return tableArn
    }
}
```

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

Delete a table

The following code example shows how to delete a DynamoDB table.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteDynamoDBTable(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 DeleteTable in *AWS SDK for Kotlin API reference*.

Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun deleteDymamoDBItem(tableNameVal: String, keyName: String, keyVal: String) {

    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request = DeleteItemRequest {
        tableName = tableNameVal
        key = keyToGet
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

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

Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun getSpecificItem(tableNameVal: String, keyName: String, keyVal: String) {

    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request = GetItemRequest {
        key = keyToGet
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

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

List tables

The following code example shows how to list DynamoDB tables.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listAllTables() {

    DynamoDbClient { 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*.

Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun putItemInTable(
    tableNameVal: String,
    key: String,
    keyVal: String,
    albumTitle: String,
    albumTitleValue: String,
    awards: String,
    awardVal: String,
    songTitle: String,
    songTitleVal: String
) {
    val itemValues = mutableMapOf<String, AttributeValue>()

    // Add all content to the table.
    itemValues[key] = AttributeValue.S(keyVal)
    itemValues[songTitle] = AttributeValue.S(songTitleVal)
    itemValues[albumTitle] = AttributeValue.S(albumTitleValue)
    itemValues[awards] = AttributeValue.S(awardVal)

    val request = PutItemRequest {
        tableName = tableNameVal
        item = itemValues
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
```

```
}
```

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

Query a table

The following code example shows how to query a DynamoDB table.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun queryDynTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionKeyVal: String,
    partitionAlias: String
): Int {

    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = partitionKeyName

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.S(partitionKeyVal)

    val request = QueryRequest {
        tableName = tableNameVal
        keyConditionExpression = "$partitionAlias = :$partitionKeyName"
        expressionAttributeNames = attrNameAlias
        this.expressionAttributeValues = attrValues
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

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

Scan a table

The following code example shows how to scan a DynamoDB table.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun scanItems(tableNameVal: String) {
```

```
    val request = ScanRequest {
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

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

Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun updateTableItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
    name: String,
    updateVal: String
) {

    val itemKey = mutableMapOf<String, AttributeValue>()
    itemKey[keyName] = AttributeValue.S(keyVal)

    val updatedValues = mutableMapOf<String, AttributeValueUpdate>()
    updatedValues[name] = AttributeValueUpdate {
        value = AttributeValue.S(updateVal)
        action = AttributeAction.Put
    }

    val request = UpdateItemRequest {
        tableName = tableNameVal
        key = itemKey
        attributeUpdates = updatedValues
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
}
```

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

Scenarios

Get started using tables, items, and queries

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.
- Delete the table.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

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 provisionedVal = ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }

    val request = CreateTableRequest {
        attributeDefinitions = listOf(attDef, attDef1)
        keySchema = listOf(keySchemaVal, keySchemaVal1)
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
```

```
        ddb.waitUntilTableExists { // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}
```

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 { 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 { 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(args: Array<String>) {

    val usage = """
        Usage:
          <fileName>

        Where:
           fileName - The path to the moviedata.json you can download from the Amazon
 DynamoDB Developer Guide.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    // Get the moviedata.json from the Amazon DynamoDB Developer Guide.
    val tableName = "Movies"
    val fileName = args[0]
    val partitionAlias = "#a"

    println("Creating an Amazon DynamoDB table named Movies with a key named id and a
 sort key named title.")
    createScenarioTable(tableName, "year")
    loadData(tableName, fileName)
    getMovie(tableName, "year", "1933")
    scanMovies(tableName)
    val count = queryMovieTable(tableName, "year", partitionAlias)
    println("There are $count Movies released in 2013.")
    deletIssuesTable(tableName)
}

suspend fun createScenarioTable(tableNameVal: String, key: String) {

    val attDef = AttributeDefinition {
        attributeName = key
        attributeType = ScalarAttributeType.N
    }

    val attDef1 = AttributeDefinition {
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }
```

```
    val keySchemaVal = KeySchemaElement {
        attributeName = key
        keyType = KeyType.Hash
    }

    val keySchemaVal1 = KeySchemaElement {
        attributeName = "title"
        keyType = KeyType.Range
    }

    val provisionedVal = ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }

    val request = CreateTableRequest {
        attributeDefinitions = listOf(attDef, attDef1)
        keySchema = listOf(keySchemaVal, keySchemaVal1)
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
        ddb.waitUntilTableExists { // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}

// Load data into the table.
suspend fun loadData(tableName: String, fileName: String) {

    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {

        if (t == 50)
            break

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
```

```
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request = PutItemRequest {
        tableName = tableNameVal
        item = itemValues
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}

suspend fun getMovie(tableNameVal: String, keyName: String, keyVal: String) {

    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request = GetItemRequest {
        key = keyToGet
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}

suspend fun deletIssuesTable(tableNameVal: String) {

    val request = DeleteTableRequest {
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun queryMovieTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionAlias: String
): Int {

    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = "year"

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.N("2013")

    val request = QueryRequest {
        tableName = tableNameVal
        keyConditionExpression = "$partitionAlias = :$partitionKeyName"
        expressionAttributeNames = attrNameAlias
        this.expressionAttributeValues = attrValues
```

```
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}

suspend fun scanMovies(tableNameVal: String) {

    val request = ScanRequest {
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

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**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun main() {
```

```
    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQBatch"
    println("Creating an Amazon DynamoDB table named $tableName with a key named id and
 a sort key named title.")
    createTablePartiQLBatch(ddb, tableName, "year")
    putRecordBatch(ddb)
    updateTableItemBatchBatch(ddb)
    deleteItemsBatch(ddb)
    deleteTablePartiQLBatch(tableName)
}

suspend fun createTablePartiQLBatch(ddb: DynamoDbClient, tableNameVal: String, key:
 String) {

    val attDef = AttributeDefinition {
        attributeName = key
        attributeType = ScalarAttributeType.N
    }

    val attDef1 = AttributeDefinition {
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }

    val keySchemaVal = KeySchemaElement {
        attributeName = key
        keyType = KeyType.Hash
    }

    val keySchemaVal1 = KeySchemaElement {
        attributeName = "title"
        keyType = KeyType.Range
    }

    val provisionedVal = ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }

    val request = CreateTableRequest {
        attributeDefinitions = listOf(attDef, attDef1)
        keySchema = listOf(keySchemaVal, keySchemaVal1)
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists { // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun putRecordBatch(ddb: DynamoDbClient) {

    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?, 'title' : ?,
 'info' : ?}"

    // Create three movies to add to the Amazon DynamoDB table.
    val parametersMovie1 = mutableListOf<AttributeValue>()
    parametersMovie1.add(AttributeValue.N("2022"))
    parametersMovie1.add(AttributeValue.S("My Movie 1"))
    parametersMovie1.add(AttributeValue.S("No Information"))
```

```
    val statementRequestMovie1 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersMovie1
    }

    // Set data for Movie 2.
    val parametersMovie2 = mutableListOf<AttributeValue>()
    parametersMovie2.add(AttributeValue.N("2022"))
    parametersMovie2.add(AttributeValue.S("My Movie 2"))
    parametersMovie2.add(AttributeValue.S("No Information"))

    val statementRequestMovie2 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersMovie2
    }

    // Set data for Movie 3.
    val parametersMovie3 = mutableListOf<AttributeValue>()
    parametersMovie3.add(AttributeValue.N("2022"))
    parametersMovie3.add(AttributeValue.S("My Movie 3"))
    parametersMovie3.add(AttributeValue.S("No Information"))

    val statementRequestMovie3 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersMovie3
    }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestMovie1)
    myBatchStatementList.add(statementRequestMovie2)
    myBatchStatementList.add(statementRequestMovie3)

    val batchRequest = BatchExecuteStatementRequest {
        statements = myBatchStatementList
    }
    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: " + response.toString())
    println("Added new movies using a batch command.")
}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
        "UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack' where year=? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
    val statementRequestRec1 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec1
    }

    // Update record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec2
    }

    // Update record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
```

```
    val statementRequestRec3 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec3
    }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest = BatchExecuteStatementRequest {
        statements = myBatchStatementList
    }

    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: $response")
    println("Updated three movies using a batch command.")
    println("Items were updated!")
}

suspend fun deleteItemsBatch(ddb: DynamoDbClient) {

    // Specify three records to delete.
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))

    val statementRequestRec1 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec1
    }

    // Specify record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec2
    }

    // Specify record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec3
    }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest = BatchExecuteStatementRequest {
        statements = myBatchStatementList
    }

    ddb.batchExecuteStatement(batchRequest)
    println("Deleted three movies using a batch command.")
}
```

```
suspend fun deleteTablePartiQLBatch(tableNameVal: String) {

    val request = DeleteTableRequest {
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- 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**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
          <fileName>

        Where:
           fileName - The path to the moviedata.json you can download from the Amazon
 DynamoDB Developer Guide.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQ"

    // Get the moviedata.json from the Amazon DynamoDB Developer Guide.
    val fileName = args[0]
    println("Creating an Amazon DynamoDB table named MoviesPartiQ with a key named id
 and a sort key named title.")
    createTablePartiQL(ddb, tableName, "year")
    loadDataPartiQL(ddb, fileName)

    println("******* Getting data from the MoviesPartiQ table.")
```

```
    getMoviePartiQL(ddb)

    println("******* Putting a record into the MoviesPartiQ table.")
    putRecordPartiQL(ddb)

    println("******* Updating a record.")
    updateTableItemPartiQL(ddb)

    println("******* Querying the movies released in 2013.")
    queryTablePartiQL(ddb)

    println("******* Deleting the MoviesPartiQ table.")
    deleteTablePartiQL(tableName)
}

suspend fun createTablePartiQL(ddb: DynamoDbClient, tableNameVal: String, key: String)
 {

    val attDef = AttributeDefinition {
        attributeName = key
        attributeType = ScalarAttributeType.N
    }

    val attDef1 = AttributeDefinition {
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }

    val keySchemaVal = KeySchemaElement {
        attributeName = key
        keyType = KeyType.Hash
    }

    val keySchemaVal1 = KeySchemaElement {
        attributeName = "title"
        keyType = KeyType.Range
    }

    val provisionedVal = ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }

    val request = CreateTableRequest {
        attributeDefinitions = listOf(attDef, attDef1)
        keySchema = listOf(keySchemaVal, keySchemaVal1)
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists { // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun loadDataPartiQL(ddb: DynamoDbClient, fileName: String) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
    var t = 0
```

```
    while (iter.hasNext()) {

        if (t == 200)
            break

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
        parameters.add(AttributeValue.N(year.toString()))
        parameters.add(AttributeValue.S(title))
        parameters.add(AttributeValue.S(info))

        executeStatementPartiQL(ddb, sqlStatement, parameters)
        println("Added Movie $title")
        parameters.clear()
        t++
    }
}

suspend fun getMoviePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?"
    val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
    parameters.add(AttributeValue.N("2012"))
    parameters.add(AttributeValue.S("The Perks of Being a Wallflower"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun putRecordPartiQL(ddb: DynamoDbClient) {

    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2020"))
    parameters.add(AttributeValue.S("My Movie"))
    parameters.add(AttributeValue.S("No Info"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Added new movie.")
}

suspend fun updateTableItemPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C. Cooper
\",\"Ernest B. Schoedsack\' where year=? and title=?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    parameters.add(AttributeValue.S("The East"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Item was updated!")
}

// Query the table where the year is 2013.
suspend fun queryTablePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year = ?"

    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun deleteTablePartiQL(tableNameVal: String) {
```

```
    val request = DeleteTableRequest {
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun executeStatementPartiQL(
    ddb: DynamoDbClient,
    statementVal: String,
    parametersVal: List<AttributeValue>
): ExecuteStatementResponse {

    val request = ExecuteStatementRequest {
        statement = statementVal
        parameters = parametersVal
    }

    return ddb.executeStatement(request)
}
```

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

Amazon EC2 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 Amazon EC2 Auto Scaling.

Actions are code excerpts that show you how to call individual Amazon EC2 Auto Scaling functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon EC2 Auto Scaling functions.

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

Topics

Scenarios

Manage groups and instances

The following code example shows how to:

- Create an Amazon EC2 Auto Scaling group and configure it with a launch template and Availability Zones.
- Get information about the group and running instances.
- Enable Amazon CloudWatch metrics collection on the group.
- Update the desired capacity of the group 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 that are collected during the example.
- Stop collecting metrics, terminate all instances, and delete the group.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

> **Tip**
> To learn how to set up and run this example, see GitHub.

```
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")
    getSpecificAutoScalingGroups(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 getSpecificAutoScalingGroups(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

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.

Actions are code excerpts that show you how to call individual EventBridge functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple EventBridge functions.

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

Topics

Actions

Create a scheduled rule

The following code example shows how to create an Amazon EventBridge scheduled rule.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createEBRule(ruleNameVal: String) {

    val request = PutRuleRequest {
        name = ruleNameVal
        eventBusName = "default"
        eventPattern = "{\"source\":[\"aws.s3\"],\"detail-type\":[\"AWS API Call
 via CloudTrail\"],\"detail\":{\"eventSource\":[\"s3.amazonaws.com\"],\"eventName\":
[\"DeleteBucket\"]}}"
        description = "A test rule created by the AWS SDK for Kotlin"
    }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(request)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

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

Delete a scheduled rule

The following code example shows how to delete an Amazon EventBridge scheduled rule.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteEBRule(ruleName: String) {

    val request = DisableRuleRequest {
        name = ruleName
        eventBusName = "default"
    }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        eventBrClient.disableRule(request)
        val ruleRequest = DeleteRuleRequest {
            name = ruleName
            eventBusName = "default"
        }

        eventBrClient.deleteRule(ruleRequest)
        println("Rule $ruleName was successfully deleted!")
    }
}
```

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

Send events

The following code example shows how to send Amazon EventBridge events.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun putEBEvents(resourceArn: String, resourceArn2: String) {

    // Populate a List with the resource ARN values.
    val resourcesOb = mutableListOf<String>()
    resourcesOb.add(resourceArn)
    resourcesOb.add(resourceArn2)

    val reqEntry = PutEventsRequestEntry {
        resources = resourcesOb
        source = "com.mycompany.myapp"
        detailType = "myDetailType"
        detail = "{ \"key1\": \"value1\", \"key2\": \"value2\" }"
    }
```

```
    val request = PutEventsRequest {
        entries = listOf(reqEntry)
    }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val response = eventBrClient.putEvents(request)
        response.entries?.forEach { resultEntry ->

            if (resultEntry.eventId != null) {
                println("Event Id is ${resultEntry.eventId}")
            } else {
                println("Injection failed with Error Code ${resultEntry.errorCode}")
            }
        }
    }
}
```

- For API details, see PutEvents 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.

Actions are code excerpts that show you how to call individual AWS Glue functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple AWS Glue functions.

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

Topics

Actions

Create a crawler

The following code example shows how to create an AWS Glue crawler.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String
```

```
) {
    val s3Target = S3Target {
        path = s3Path
    }

    // Add the S3Target to a list.
    val targetList = mutableListOf<S3Target>()
    targetList.add(s3Target)

    val targetOb = CrawlerTargets {
        s3Targets = targetList
    }

    val request = CreateCrawlerRequest {
        databaseName = dbName
        name = crawlerName
        description = "Created by the AWS Glue Kotlin API"
        targets = targetOb
        role = iam
        schedule = cron
    }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

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

Get a crawler

The following code example shows how to get an AWS Glue crawler.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun getSpecificCrawler(crawlerName: String?) {

    val request = GetCrawlerRequest {
        name = crawlerName
    }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

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

Get a database from the Data Catalog

The following code example shows how to get a database from the AWS Glue Data Catalog.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun getSpecificDatabase(databaseName: String?) {

    val request = GetDatabaseRequest {
        name = databaseName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

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

Start a crawler

The following code example shows how to start an AWS Glue crawler.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun startSpecificCrawler(crawlerName: String?) {

    val request = StartCrawlerRequest {
        name = crawlerName
    }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

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

Scenarios

Get started running crawlers and jobs

The following code example shows how to:

- Create and run a crawler that crawls a public Amazon Simple Storage Service (Amazon S3) bucket and generates a metadata database that describes the CSV-formatted data it finds.

- List information about databases and tables in your AWS Glue Data Catalog.
- Create and run a job that extracts CSV data from the source Amazon S3 bucket, transforms it by removing and renaming fields, and loads JSON-formatted output into another Amazon S3 bucket.
- List information about job runs and view some of the transformed data.
- Delete all resources created by the demo.

For more information, see Tutorial: Getting started with AWS Glue Studio.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName> <scriptLocation>
 <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
 Management (IAM) role that has AWS Glue and Amazon Simple Storage Service (Amazon S3)
 permissions.
            s3Path - The Amazon Simple Storage Service (Amazon S3) target that contains
 data (for example, CSV data).
            cron - A cron expression used to specify the schedule (for example, cron(15
 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
    }

    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
    val jobName = args[5]
    val scriptLocation = args[6]
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
```

```
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(dbName: String?, locationUriVal: String?) {

    val input = DatabaseInput {
        description = "Built with the AWS SDK for Kotlin"
        name = dbName
        locationUri = locationUriVal
    }

    val request = CreateDatabaseRequest {
        databaseInput = input
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}

suspend fun createCrawler(iam: String?, s3Path: String?, cron: String?, dbName:
 String?, crawlerName: String) {

    val s3Target = S3Target {
        path = s3Path
    }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb = CrawlerTargets {
        s3Targets = targetList
    }

    val crawlerRequest = CreateCrawlerRequest {
        databaseName = dbName
        name = crawlerName
        description = "Created by the AWS Glue Java API"
        targets = targetOb
        role = iam
        schedule = cron
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {

    val request = GetCrawlerRequest {
        name = crawlerName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
```

```
}

suspend fun startCrawler(crawlerName: String) {

    val crawlerRequest = StartCrawlerRequest {
        name = crawlerName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {

    val request = GetDatabaseRequest {
        name = databaseName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {

    val tableRequest = GetTablesRequest {
        databaseName = dbName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}

suspend fun startJob(jobNameVal: String?) {

    val runRequest = StartJobRunRequest {
        workerType = WorkerType.G1X
        numberOfWorkers = 10
        jobName = jobNameVal
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(jobName: String, iam: String?, scriptLocationVal: String?) {

    val commandOb = JobCommand {
        pythonVersion = "3"
        name = "MyJob1"
        scriptLocation = scriptLocationVal
    }

    val jobRequest = CreateJobRequest {
        description = "A Job created by using the AWS SDK for Java V2"
        glueVersion = "2.0"
```

```
        workerType = WorkerType.G1X
        numberOfWorkers = 10
        name = jobName
        role = iam
        command = commandOb
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {

    val request = GetJobsRequest {
        maxResults = 10
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {

    val request = GetJobRunsRequest {
        jobName = jobNameVal
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {

    val jobRequest = DeleteJobRequest {
        jobName = jobNameVal
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {

    val request = DeleteDatabaseRequest {
        name = databaseName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
```

```
    val request = DeleteCrawlerRequest {
        name = crawlerName
    }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

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.

Actions are code excerpts that show you how to call individual IAM functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple IAM functions.

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

Topics

Actions

Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun attachIAMRolePolicy(roleNameVal: String, policyArnVal: String) {

    val request = ListAttachedRolePoliciesRequest {
        roleName = roleNameVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1)
                return
        }

        val policyRequest = AttachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkList(attachedPolicies: List<AttachedPolicy>, policyArnVal: String): Int {

    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

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

Create a policy

The following code example shows how to create an IAM policy.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun createIAMPolicy(policyNameVal: String?): String {

    val policyDocumentVal = "{" +
```

```
        "  \"Version\": \"2012-10-17\"," +
        "  \"Statement\": [" +
        "    {" +
        "        \"Effect\": \"Allow\"," +
        "        \"Action\": [" +
        "            \"dynamodb:DeleteItem\"," +
        "            \"dynamodb:GetItem\"," +
        "            \"dynamodb:PutItem\"," +
        "            \"dynamodb:Scan\"," +
        "            \"dynamodb:UpdateItem\"" +
        "        ]," +
        "        \"Resource\": \"*\"" +
        "    }" +
        "   ]" +
        "}"

    val request = CreatePolicyRequest {
        policyName = policyNameVal
        policyDocument = policyDocumentVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

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

Create a user

The following code example shows how to create an IAM user.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createIAMUser(usernameVal: String?): String? {

    val request = CreateUserRequest {
        userName = usernameVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

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

Create an access key

The following code example shows how to create an IAM access key.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createIAMAccessKey(user: String?): String {

    val request = CreateAccessKeyRequest {
        userName = user
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

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

Create an alias for an account

The following code example shows how to create an alias for an IAM account.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createIAMAccountAlias(alias: String) {

    val request = CreateAccountAliasRequest {
        accountAlias = alias
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

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

Delete a policy

The following code example shows how to delete an IAM policy.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {

    val request = DeletePolicyRequest {
        policyArn = policyARNVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

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

Delete a user

The following code example shows how to delete an IAM user.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteIAMUser(userNameVal: String) {

    val request = DeleteUserRequest {
        userName = userNameVal
    }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

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

Delete an access key

The following code example shows how to delete an IAM access key.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteKey(userNameVal: String, accessKey: String) {

    val request = DeleteAccessKeyRequest {
```

```
        accessKeyId = accessKey
        userName = userNameVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

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

Delete an account alias

The following code example shows how to delete an IAM account alias.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteIAMAccountAlias(alias: String) {

    val request = DeleteAccountAliasRequest {
        accountAlias = alias
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

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

Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun detachPolicy(roleNameVal: String, policyArnVal: String) {

    val request = DetachRolePolicyRequest {
        roleName = roleNameVal
        policyArn = policyArnVal
    }
```

```
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
}
```

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

Get a policy

The following code example shows how to get an IAM policy.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun getIAMPolicy(policyArnVal: String?) {

    val request = GetPolicyRequest {
        policyArn = policyArnVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

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

List a user's access keys

The following code example shows how to list a user's IAM access keys.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listKeys(userNameVal: String?) {

    val request = ListAccessKeysRequest {
        userName = userNameVal
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
```

```
}
```

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

List account aliases

The following code example shows how to list IAM account aliases.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listAliases() {

    IamClient { 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*.

List users

The following code example shows how to list IAM users.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listAllUsers() {

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null)
                println("Permissions boundary details
 ${permissionsBoundary.permissionsBoundaryType}")
        }
    }
}
```

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

Update a user

The following code example shows how to update an IAM user.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun updateIAMUser(curName: String?, newName: String?) {

    val request = UpdateUserRequest {
        userName = curName
        newUserName = newName
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

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

Scenarios

Create a user and assume a role

The following code example shows how to:

- Create a user who has 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 Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

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

```
        "   ]" +
        "}"

    val request = CreatePolicyRequest {
        policyName = policyNameVal
        policyDocument = policyDocumentValue
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(rolenameVal: String?, fileLocation: String?): String? {

    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request = CreateRoleRequest {
        roleName = rolenameVal
        assumeRolePolicyDocument = jsonObject.toJSONString()
        description = "Created using the AWS SDK for Kotlin"
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(roleNameVal: String, policyArnVal: String) {

    val request = ListAttachedRolePoliciesRequest {
        roleName = roleNameVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1)
                return
        }

        val policyRequest = AttachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkMyList(attachedPolicies: List<AttachedPolicy>, policyArnVal: String): Int {

    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
```

```
    }
    return 0
}

suspend fun assumeGivenRole(roleArnVal: String?, roleSessionNameVal: String?,
 bucketName: String) {

    val stsClient = StsClient {
        region = "us-east-1"
    }

    val roleRequest = AssumeRoleRequest {
        roleArn = roleArnVal
        roleSessionName = roleSessionNameVal
    }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials = StaticCredentialsProvider {
        accessKeyId = key
        secretAccessKey = secKey
        sessionToken = secToken
    }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 = S3Client {
        credentialsProvider = staticCredentials
        region = "us-east-1"
    }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects = ListObjectsRequest {
        bucket = bucketName
    }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(roleNameVal: String, polArn: String) {

    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest = DetachRolePolicyRequest {
        policyArn = polArn
        roleName = roleNameVal
    }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request = DeletePolicyRequest {
        policyArn = polArn
    }

    iam.deletePolicy(request)
```

```
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest = DeleteRoleRequest {
        roleName = roleNameVal
    }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request = DeleteUserRequest {
        userName = userNameVal
    }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- 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

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 that show you how to call individual AWS KMS functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple AWS KMS functions.

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

Topics

Actions

Create a grant for a key

The following code example shows how to create a grant for a KMS key.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createNewGrant(keyIdVal: String?, granteePrincipalVal: String?, operation:
 String): String? {

    val operationOb = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationOb)

    val request = CreateGrantRequest {
        keyId = keyIdVal
        granteePrincipal = granteePrincipalVal
        operations = grantOperationList
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

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

Create a key

The following code example shows how to create an AWS KMS key.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createKey(keyDesc: String?): String? {

    val request = CreateKeyRequest {
        description = keyDesc
        customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
        keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
```

```
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

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

Create an alias for a key

The following code example shows how to create an alias for a KMS key key.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createCustomAlias(targetKeyIdVal: String?, aliasNameVal: String?) {

    val request = CreateAliasRequest {
        aliasName = aliasNameVal
        targetKeyId = targetKeyIdVal
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

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

Decrypt ciphertext

The following code example shows how to decrypt ciphertext that was encrypted by a KMS key.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {

    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest = EncryptRequest {
        keyId = keyIdValue
        plaintext = myBytes
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
```

```
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(encryptedDataVal: ByteArray?, keyIdVal: String?, path: String)
 {

    val decryptRequest = DecryptRequest {
        ciphertextBlob = encryptedDataVal
        keyId = keyIdVal
    }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Write the decrypted data to a file.
        if (myVal != null) {
            File(path).writeBytes(myVal)
        }
    }
}
```

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

Describe a key

The following code example shows how to describe a KMS key.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun describeSpecifcKey(keyIdVal: String?) {

    val request = DescribeKeyRequest {
        keyId = keyIdVal
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

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

Disable a key

The following code example shows how to disable a KMS key.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun disableKey(keyIdVal: String?) {

    val request = DisableKeyRequest {
        keyId = keyIdVal
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

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

Enable a key

The following code example shows how to enable a KMS key.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun enableKey(keyIdVal: String?) {

    val request = EnableKeyRequest {
        keyId = keyIdVal
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

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

Encrypt text using a key

The following code example shows how to encrypt text using a KMS key.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {

    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest = EncryptRequest {
        keyId = keyIdValue
        plaintext = myBytes
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(encryptedDataVal: ByteArray?, keyIdVal: String?, path: String)
 {

    val decryptRequest = DecryptRequest {
        ciphertextBlob = encryptedDataVal
        keyId = keyIdVal
    }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Write the decrypted data to a file.
        if (myVal != null) {
            File(path).writeBytes(myVal)
        }
    }
}
```

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

List aliases for a key

The following code example shows how to list aliases for a KMS key.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listAllAliases() {

    val request = ListAliasesRequest {
        limit = 15
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
```

```
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

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

List grants for a key

The following code example shows how to list grants for a KMS key.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun displayGrantIds(keyIdVal: String?) {

    val request = ListGrantsRequest {
        keyId = keyIdVal
        limit = 15
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

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

List keys

The following code example shows how to list KMS keys.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listAllKeys() {

    val request = ListKeysRequest {
        limit = 15
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
```

```
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- 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.

Actions are code excerpts that show you how to call individual Lambda functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Lambda functions.

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

Topics

Actions

Create a function

The following code example shows how to create a Lambda function.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createNewFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String
): String? {

    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
```

```
        runtime = Runtime.Java8
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn
    }
}
```

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

Delete a function

The following code example shows how to delete a Lambda function.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun delLambdaFunction(myFunctionName: String) {

    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

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

Invoke a function

The following code example shows how to invoke a Lambda function.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
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

Get started with functions

The following code example shows how to:

- Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.
- Create a Lambda function and upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure its Lambda environment with an environment variable.
- Invoke the function with new parameters and get results. Display the execution log that's returned from the invocation.
- List the functions for your account.
- Delete the IAM role and the Lambda function.

For more information, see Create a Lambda function with the console.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
            <functionName> <role> <handler> <bucketName> <updatedBucketName> <key>

        Where:
            functionName - The name of the AWS Lambda function.
            role - The AWS Identity and Access Management (IAM) service role that has
 AWS Lambda permissions.
            handler - The fully qualified method name (for example,
 example.Handler::handleRequest).
            bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name that
 contains the ZIP or JAR used for the Lambda function's code.
            updatedBucketName - The Amazon S3 bucket name that contains the .zip
 or .jar used to update the Lambda function's code.
            key - The Amazon S3 key name that represents the .zip or .jar file (for
 example, LambdaHello-1.0-SNAPSHOT.jar).
```

```
            """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val functionName = args[0]
    val role = args[1]
    val handler = args[2]
    val bucketName = args[3]
    val updatedBucketName = args[4]
    val key = args[5]

    println("Creating a Lambda function named $functionName.")
    val funArn = createScFunction(functionName, bucketName, key, handler, role)
    println("The AWS Lambda ARN is $funArn")

    // Get a specific Lambda function.
    println("Getting the $functionName AWS Lambda function.")
    getFunction(functionName)

    // List the Lambda functions.
    println("Listing all AWS Lambda functions.")
    listFunctionsSc()

    // Invoke the Lambda function.
    println("*** Invoke the Lambda function.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function code.
    println("*** Update the Lambda function code.")
    updateFunctionCode(functionName, updatedBucketName, key)

    // println("*** Invoke the function again after updating the code.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function configuration.
    println("Update the run time of the function.")
    UpdateFunctionConfiguration(functionName, handler)

    // Delete the AWS Lambda function.
    println("Delete the AWS Lambda function.")
    delFunction(functionName)
}

suspend fun createScFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String
): String {

    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java8
```

```
    }

    // Create a Lambda function using a waiter
    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn.toString()
    }
}

suspend fun getFunction(functionNameVal: String) {

    val functionRequest = GetFunctionRequest {
        functionName = functionNameVal
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val response = awsLambda.getFunction(functionRequest)
        println("The runtime of this Lambda function is
 ${response.configuration?.runtime}")
    }
}

suspend fun listFunctionsSc() {

    val request = ListFunctionsRequest {
        maxItems = 10
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val response = awsLambda.listFunctions(request)
        response.functions?.forEach { function ->
            println("The function name is ${function.functionName}")
        }
    }
}

suspend fun invokeFunctionSc(functionNameVal: String) {

    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request = InvokeRequest {
        functionName = functionNameVal
        payload = byteArray
        logType = LogType.Tail
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("The function payload is ${res.payload?.toString(Charsets.UTF_8)}")
    }
}

suspend fun updateFunctionCode(functionNameVal: String?, bucketName: String?, key:
 String?) {

    val functionCodeRequest = UpdateFunctionCodeRequest {
        functionName = functionNameVal
        publish = true
        s3Bucket = bucketName
        s3Key = key
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
```

```
        val response = awsLambda.updateFunctionCode(functionCodeRequest)
        awsLambda.waitUntilFunctionUpdated {
            functionName = functionNameVal
        }
        println("The last modified value is " + response.lastModified)
    }
}

suspend fun UpdateFunctionConfiguration(functionNameVal: String?, handlerVal: String?)
 {

    val configurationRequest = UpdateFunctionConfigurationRequest {
        functionName = functionNameVal
        handler = handlerVal
        runtime = Runtime.Java11
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.updateFunctionConfiguration(configurationRequest)
    }
}

suspend fun delFunction(myFunctionName: String) {

    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

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 that show you how to call individual Amazon Pinpoint functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon Pinpoint functions.

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

Topics

Actions

Create a campaign

The following code example shows how to create a campaign.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun createPinCampaign(appId: String, segmentIdVal: String) {

    val scheduleOb = Schedule {
        startTime = "IMMEDIATE"
    }

    val defaultMessageOb = Message {
        action = Action.OpenApp
        body = "My message body"
        title = "My message title"
    }

    val messageConfigurationOb = MessageConfiguration {
        defaultMessage = defaultMessageOb
    }

    val writeCampaign = WriteCampaignRequest {
        description = "My description"
        schedule = scheduleOb
        name = "MyCampaign"
        segmentId = segmentIdVal
        messageConfiguration = messageConfigurationOb
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse = pinpoint.createCampaign(
            CreateCampaignRequest {
                applicationId = appId
                writeCampaignRequest = writeCampaign
            }
        )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

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

Create a segment

The following code example shows how to create a segment.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun createPinpointSegment(applicationIdVal: String?): String? {

    val segmentAttributes = mutableMapOf<String, AttributeDimension>()
    val myList = mutableListOf<String>()
    myList.add("Lakers")

    val atts = AttributeDimension {
        attributeType = AttributeType.Inclusive
        values = myList
    }

    segmentAttributes["Team"] = atts
    val recencyDimension = RecencyDimension {
        duration = Duration.fromValue("DAY_30")
        recencyType = RecencyType.fromValue("ACTIVE")
    }

    val segmentBehaviors = SegmentBehaviors {
        recency = recencyDimension
    }

    val segmentLocation = SegmentLocation {}
    val dimensionsOb = SegmentDimensions {
        attributes = segmentAttributes
        behavior = segmentBehaviors
        demographic = SegmentDemographics {}
        location = segmentLocation
    }

    val writeSegmentRequestOb = WriteSegmentRequest {
        name = "MySegment101"
        dimensions = dimensionsOb
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse = pinpoint.createSegment(
            CreateSegmentRequest {
                applicationId = applicationIdVal
                writeSegmentRequest = writeSegmentRequestOb
            }
        )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}
```

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

Create an application

The following code example shows how to create an appliation.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createApplication(applicationName: String?): String? {
```

```
    val createApplicationRequestOb = CreateApplicationRequest {
        name = applicationName
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.createApp(
            CreateAppRequest {
                createApplicationRequest = createApplicationRequestOb
            }
        )
        return result.applicationResponse?.id
    }
}
```

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

Delete an application

The following code example shows how to delete an application.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deletePinApp(appId: String?) {

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteApp(
            DeleteAppRequest {
                applicationId = appId
            }
        )
        val appName = result.applicationResponse?.name
        println("Application $appName has been deleted.")
    }
}
```

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

Delete an endpoint

The following code example shows how to delete an endpoint.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deletePinEncpoint(appIdVal: String?, endpointIdVal: String?) {

    val deleteEndpointRequest = DeleteEndpointRequest {
```

```
        applicationId = appIdVal
        endpointId = endpointIdVal
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is  $id")
    }
}
```

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

Get endpoints

The following code example shows how to get endpoints.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun lookupPinpointEndpoint(appId: String?, endpoint: String?) {

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.getEndpoint(
            GetEndpointRequest {
                applicationId = appId
                endpointId = endpoint
            }
        )
        val endResponse = result.endpointResponse

        // Uses the Google Gson library to pretty print the endpoint JSON.
        val gson: com.google.gson.Gson = GsonBuilder()
            .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
            .setPrettyPrinting()
            .create()

        val endpointJson: String = gson.toJson(endResponse)
        println(endpointJson)
    }
}
```

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

List segments

The following code example shows how to list segments.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listSegs(appId: String?) {

    PinpointClient { region = "us-west-2" }.use { pinpoint ->

        val response = pinpoint.getSegments(
            GetSegmentsRequest {
                applicationId = appId
            }
        )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
    }
}
```

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

Send email and text messages

The following code example shows how to send email and text messages with Amazon Pinpoint.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun sendEmail(
    msgSubject: String?,
    appId: String?,
    senderAddress: String?,
    toAddress: String
) {

    // The body of the email for recipients whose email clients support HTML content.
    val htmlBody = (
        "<h1>Amazon Pinpoint test (AWS SDK for Kotlin)</h1>" +
            "<p>This email was sent through the <a href='https://aws.amazon.com/
pinpoint/'>" +
            "Amazon Pinpoint</a> Email API"
        )

    // The character encoding to use for the subject line and the message body.
    val charsetVal = "UTF-8"

    val addressMap = mutableMapOf<String, AddressConfiguration>()
    val configuration = AddressConfiguration {
        channelType = ChannelType.Email
    }

    addressMap[toAddress] = configuration
    val emailPart = SimpleEmailPart {
        data = htmlBody
        charset = charsetVal
    }

    val subjectPartOb = SimpleEmailPart {
        data = msgSubject
        charset = charsetVal
```

```
    }

    val simpleEmailOb = SimpleEmail {
        htmlPart = emailPart
        subject = subjectPartOb
    }

    val emailMessageOb = EmailMessage {
        body = htmlBody
        fromAddress = senderAddress
        simpleEmail = simpleEmailOb
    }

    val directMessageConfigurationOb = DirectMessageConfiguration {
        emailMessage = emailMessageOb
    }

    val messageRequestOb = MessageRequest {
        addresses = addressMap
        messageConfiguration = directMessageConfigurationOb
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        pinpoint.sendMessages(
            SendMessagesRequest {
                applicationId = appId
                messageRequest = messageRequestOb
            }
        )
        println("The email message was successfully sent")
    }
}
```

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

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 that show you how to call individual Amazon Rekognition functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon Rekognition functions.

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

Topics

Actions

Compare faces in an image against a reference image

The following code example shows how to compare faces in an image against a reference image with Amazon Rekognition.

For more information, see Comparing faces in images.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun compareTwoFaces(similarityThresholdVal: Float, sourceImageVal: String,
 targetImageVal: String) {

    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage = Image {
        bytes = sourceBytes
    }

    val tarImage = Image {
        bytes = targetBytes
    }

    val facesRequest = CompareFacesRequest {
        sourceImage = souImage
        targetImage = tarImage
        similarityThreshold = similarityThresholdVal
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null)
                    println("Face at ${position.left} ${position.top} matches with
 ${face.confidence} % confidence.")
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null)
            println("There was ${uncompared.size} face(s) that did not match")

        println("Source image rotation:
 ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

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

Create a collection

The following code example shows how to create an Amazon Rekognition collection.

For more information, see Creating a collection.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createMyCollection(collectionIdVal: String) {

    val request = CreateCollectionRequest {
        collectionId = collectionIdVal
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

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

Delete a collection

The following code example shows how to delete an Amazon Rekognition collection.

For more information, see Deleting a collection.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteMyCollection(collectionIdVal: String) {

    val request = DeleteCollectionRequest {
        collectionId = collectionIdVal
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

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

Delete faces from a collection

The following code example shows how to delete faces from an Amazon Rekognition collection.

For more information, see Deleting faces from a collection.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteFacesCollection(collectionIdVal: String?, faceIdVal: String) {

    val deleteFacesRequest = DeleteFacesRequest {
        collectionId = collectionIdVal
        faceIds = listOf(faceIdVal)
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

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

Describe a collection

The following code example shows how to describe an Amazon Rekognition collection.

For more information, see Describing a collection.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun describeColl(collectionName: String) {

    val request = DescribeCollectionRequest {
        collectionId = collectionName
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

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

Detect faces in an image

The following code example shows how to detect faces in an image with Amazon Rekognition.

For more information, see Detecting faces in an image.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun detectFacesinImage(sourceImage: String?) {

    val souImage = Image {
        bytes = (File(sourceImage).readBytes())
    }

    val request = DetectFacesRequest {
        attributes = listOf(Attribute.All)
        image = souImage
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between ${ageRange?.low} and
 ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

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

Detect labels in an image

The following code example shows how to detect labels in an image with Amazon Rekognition.

For more information, see Detecting labels in an image.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun detectImageLabels(sourceImage: String) {

    val souImage = Image {
        bytes = (File(sourceImage).readBytes())
    }
    val request = DetectLabelsRequest {
        image = souImage
        maxLabels = 10
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
```

```
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

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

Detect moderation labels in an image

The following code example shows how to detect moderation labels in an image with Amazon Rekognition. Moderation labels identify content that may be inappropriate for some audiences.

For more information, see Detecting inappropriate images.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun detectModLabels(sourceImage: String) {

    val myImage = Image {
        this.bytes = (File(sourceImage).readBytes())
    }

    val request = DetectModerationLabelsRequest {
        image = myImage
        minConfidence = 60f
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} % Parent:
 ${label.parentName}")
        }
    }
}
```

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

Detect text in an image

The following code example shows how to detect text in an image with Amazon Rekognition.

For more information, see Detecting text in an image.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun detectTextLabels(sourceImage: String?) {

    val souImage = Image {
        bytes = (File(sourceImage).readBytes())
    }

    val request = DetectTextRequest {
        image = souImage
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

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

Index faces to a collection

The following code example shows how to index faces in an image and add them to an Amazon Rekognition collection.

For more information, see Adding faces to a collection.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun addToCollection(collectionIdVal: String?, sourceImage: String) {

    val souImage = Image {
        bytes = (File(sourceImage).readBytes())
    }

    val request = IndexFacesRequest {
        collectionId = collectionIdVal
        image = souImage
        maxFaces = 1
        qualityFilter = QualityFilter.Auto
        detectionAttributes = listOf(Attribute.Default)
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
```

```
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")

            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

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

List collections

The following code example shows how to list Amazon Rekognition collections.

For more information, see Listing collections.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listAllCollections() {

    val request = ListCollectionsRequest {
        maxResults = 10
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

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

List faces in a collection

The following code example shows how to list faces in an Amazon Rekognition collection.

For more information, see Listing faces in a collection.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun listFacesCollection(collectionIdVal: String?) {

    val request = ListFacesRequest {
        collectionId = collectionIdVal
        maxResults = 10
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
    }
}
```

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

Recognize celebrities in an image

The following code example shows how to recognize celebrities in an image with Amazon Rekognition.

For more information, see Recognizing celebrities in an image.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {

    val souImage = Image {
        bytes = (File(sourceImage).readBytes())
    }

    val request = RecognizeCelebritiesRequest {
        image = souImage
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

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

Scenarios

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**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

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 { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {

    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest = GetFaceDetectionRequest {
            jobId = startJobId
            maxResults = 10
        }

        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
```

```
            status = response.jobStatus.toString()
            if (status.compareTo("SUCCEEDED") == 0)
                finished = true
            else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}
```

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 { 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

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.

Actions are code excerpts that show you how to call individual Amazon S3 functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon S3 functions.

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

Topics

Actions

Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String
) {

    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request = CopyObjectRequest {
        copySource = encodedUrl
        bucket = toBucket
        key = objectKey
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

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

Create a bucket

The following code example shows how to create an S3 bucket.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createNewBucket(bucketName: String) {
```

```
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { 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*.

Delete a policy from a bucket

The following code example shows how to delete a policy from an S3 bucket.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {

    val request = DeleteBucketPolicyRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

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

Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteBucketObjects(bucketName: String, objectName: String) {

    val objectId = ObjectIdentifier {
        key = objectName
    }

    val delOb = Delete {
```

```
        objects = listOf(objectId)
    }

    val request = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

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

Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun getObjectBytes(bucketName: String, keyName: String, path: String) {

    val request = GetObjectRequest {
        key = keyName
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

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

Get the ACL of an object

The following code example shows how to get the access control list (ACL) of an S3 object.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun getBucketACL(objectKey: String, bucketName: String) {

    val request = GetObjectAclRequest {
        bucket = bucketName
        key = objectKey
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

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

Get the policy for a bucket

The following code example shows how to get the policy for an S3 bucket.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun getPolicy(bucketName: String): String? {

    println("Getting policy for bucket $bucketName")

    val request = GetBucketPolicyRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

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

List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listBucketObjects(bucketName: String) {

    val request = ListObjectsRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${calKb(myObject.size)} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long {
    return intValue / 1024
}
```

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

Set a new ACL for a bucket

The following code example shows how to set a new access control list (ACL) for an S3 bucket.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun setBucketAcl(bucketName: String, idVal: String) {

    val myGrant = Grantee {
        id = idVal
        type = Type.CanonicalUser
    }

    val ownerGrant = Grant {
        grantee = myGrant
        permission = Permission.FullControl
    }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb = Owner {
        id = idVal
    }

    val acl = AccessControlPolicy {
        owner = ownerOb
        grants = grantList
    }

    val request = PutBucketAclRequest {
        bucket = bucketName
```

```
        accessControlPolicy = acl
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

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

Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun putS3Object(bucketName: String, objectKey: String, objectPath: String) {

    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        metadata = metadataVal
        body = File(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

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

Scenarios

Getting started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.

- Delete a bucket.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

> **Tip**
> To learn how to set up and run this example, see GitHub.

```
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.
    getObject(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
```

```
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(bucketName: String, objectKey: String, objectPath: String) {

    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        metadata = metadataVal
        this.body = Paths.get(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObject(bucketName: String, keyName: String, path: String) {

    val request = GetObjectRequest {
        key = keyName
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {

    val request = ListObjectsRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(fromBucket: String, objectKey: String, toBucket: String) {

    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
```

```
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request = CopyObjectRequest {
        copySource = encodedUrl
        bucket = toBucket
        key = objectKey
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(bucketName: String, objectName: String) {

    val objectId = ObjectIdentifier {
        key = objectName
    }

    val delOb = Delete {
        objects = listOf(objectId)
    }

    val request = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {

    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjects
  - PutObject

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 that show you how to call individual Amazon SNS functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon SNS functions.

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

Topics

Actions

Add tags to a topic

The following code example shows how to add tags to an Amazon SNS topic.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun addTopicTags(topicArn: String) {

    val tag = Tag {
        key = "Team"
        value = "Development"
    }

    val tag2 = Tag {
        key = "Environment"
        value = "Gamma"
    }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request = TagResourceRequest {
        resourceArn = topicArn
        tags = tagList
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

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

Create a topic

The following code example shows how to create an Amazon SNS topic.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createSNSTopic(topicName: String): String {

    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

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

Delete a subscription

The following code example shows how to delete an Amazon SNS subscription.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun unSub(subscriptionArnVal: String) {

    val request = UnsubscribeRequest {
        subscriptionArn = subscriptionArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

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

Delete a topic

The following code example shows how to delete an Amazon SNS topic and all subscriptions to that topic.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun deleteSNSTopic(topicArnVal: String) {

    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

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

Get the properties of a topic

The following code example shows how to get the properties of an Amazon SNS topic.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {

    val request = GetTopicAttributesRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

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

List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

SDK for Kotlin

Note
This is prerelease documentation for a feature in preview release. It is subject to change.

Tip
To learn how to set up and run this example, see GitHub.

```
suspend fun listSNSSubscriptions() {
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

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

List topics

The following code example shows how to list Amazon SNS topics.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listSNSTopics() {

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

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

Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun pubTextSMS(messageVal: String?, phoneNumberVal: String?) {

    val request = PublishRequest {
        message = messageVal
        phoneNumber = phoneNumberVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
```

```
        println("${result.messageId} message sent.")
    }
}
```

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

Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun pubTopic(topicArnVal: String, messageVal: String) {

    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

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

Set topic attributes

The following code example shows how to set Amazon SNS topic attributes.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun setTopAttr(attribute: String?, topicArnVal: String?, value: String?) {

    val request = SetTopicAttributesRequest {
        attributeName = attribute
        attributeValue = value
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

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

Subscribe a Lambda function to a topic

The following code example shows how to subscribe a Lambda function so it receives notifications from an Amazon SNS topic.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun subLambda(topicArnVal: String?, lambdaArn: String?) {

    val request = SubscribeRequest {
        protocol = "lambda"
        endpoint = lambdaArn
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

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

Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun subEmail(topicArnVal: String, email: String): String {

    val request = SubscribeRequest {
        protocol = "email"
        endpoint = email
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
```

```
    }
}
```

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

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 that show you how to call individual Amazon SQS functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Amazon SQS functions.

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

Topics

Actions

Create a queue

The following code example shows how to create an Amazon SQS queue.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createQueue(queueNameVal: String): String {

    println("Create Queue")
    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("Get queue url")

        val getQueueUrlRequest = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        val getQueueUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
        return getQueueUrlResponse.queueUrl.toString()
    }
}
```

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

Delete a message from a queue

The following code example shows how to delete a message from an Amazon SQS queue.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest = PurgeQueueRequest {
        queueUrl = queueUrlVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {

    val request = DeleteQueueRequest {
        queueUrl = queueUrlVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

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

Delete a queue

The following code example shows how to delete an Amazon SQS queue.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest = PurgeQueueRequest {
        queueUrl = queueUrlVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
```

```
    }
}

suspend fun deleteQueue(queueUrlVal: String) {

    val request = DeleteQueueRequest {
        queueUrl = queueUrlVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

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

List queues

The following code example shows how to list Amazon SQS queues.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest = ListQueuesRequest {
        queueNamePrefix = prefix
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
}
```

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

Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun receiveMessages(queueUrlVal: String?) {

    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest = ReceiveMessageRequest {
        queueUrl = queueUrlVal
        maxNumberOfMessages = 5
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

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

Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun sendMessages(queueUrlVal: String, message: String) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest = SendMessageRequest {
        queueUrl = queueUrlVal
        messageBody = message
        delaySeconds = 10
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessage(sendRequest)
        println("A single message was successfully sent.")
    }
}

suspend fun sendBatchMessages(queueUrlVal: String?) {
    println("Sending multiple messages")

    val msg1 = SendMessageBatchRequestEntry {
        id = "id1"
        messageBody = "Hello from msg 1"
    }

    val msg2 = SendMessageBatchRequestEntry {
        id = "id2"
        messageBody = "Hello from msg 2"
    }

    val sendMessageBatchRequest = SendMessageBatchRequest {
        queueUrl = queueUrlVal
```

```
        entries = listOf(msg1, msg2)
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

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

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 that show you how to call individual Secrets Manager functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple Secrets Manager functions.

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

Topics

Actions

Create a secret

The following code example shows how to create a Secrets Manager secret.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun createNewSecret(secretName: String?, secretValue: String?): String? {

    val request = CreateSecretRequest {
        name = secretName
        description = "This secret was created by the AWS Secrets Manager Kotlin API"
        secretString = secretValue
    }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.createSecret(request)
        return response.arn
    }
}
```

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

Describe a secret

The following code example shows how to describe a Secrets Manager secret.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun describeGivenSecret(secretName: String?) {

    val secretRequest = DescribeSecretRequest {
        secretId = secretName
    }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.describeSecret(secretRequest)
        val secArn = response.description
        println("The secret description is $secArn")
    }
}
```

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

Get a secret value

The following code example shows how to get a Secrets Manager secret value.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun getValue(secretName: String?) {

    val valueRequest = GetSecretValueRequest {
        secretId = secretName
    }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

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

List secrets

The following code example shows how to list Secrets Manager secrets.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun listAllSecrets() {

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.listSecrets(ListSecretsRequest {})
        response.secretList?.forEach { secret ->
            println("The secret name is ${secret.name}")
            println("The secret description is ${secret.description}")
        }
    }
}
```

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

Put a value in a secret

The following code example shows how to put a value in a Secrets Manager secret.

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.
>
> **Tip**
> To learn how to set up and run this example, see GitHub.

```
suspend fun updateMySecret(secretName: String?, secretValue: String?) {

    val request = UpdateSecretRequest {
        secretId = secretName
        secretString = secretValue
    }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        secretsClient.updateSecret(request)
        println("The secret value was updated")
    }
}
```

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

Cross-service examples using SDK for Kotlin

The following sample applications use the AWS SDK for Kotlin to work across multiple AWS services.

Examples

- Build an application to submit data to a DynamoDB table (p. 133)
- Build a publish and subscription application that translates messages (p. 133)

- Create a dynamic web application to track DynamoDB data (p. 133)
- Detect objects in images with Amazon Rekognition using an AWS SDK (p. 134)

Build an application to submit data to a DynamoDB table

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a native Android application that submits data using the Amazon DynamoDB Kotlin API and sends a text message using the Amazon SNS Kotlin API.

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 SNS

Build a publish and subscription application that translates messages

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

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 dynamic web application to track DynamoDB data

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

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

Detect objects in images with Amazon Rekognition using an AWS SDK

SDK for Kotlin

> **Note**
> This is prerelease documentation for a feature in preview release. It is subject to change.

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

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.

Topics

Data protection in AWS SDK for Kotlin

The shared responsibility model applies to data protection in this AWS product or service. 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. This content includes 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 user accounts with 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 recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 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) 140-2.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with this AWS product or service or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data

that you enter into this AWS product or service or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

Identity and Access Management for this AWS Product or Service

AWS Identity and Access Management (IAM) is an Amazon Web Services (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 resources in AWS services. IAM is an AWS service that you can use with no additional charge.

To use this AWS product or service to access AWS, you need an AWS account and AWS credentials. To increase the security of your AWS account, we recommend that you use an *IAM user* to provide access credentials instead of using your AWS account credentials.

For details about working with IAM, see AWS Identity and Access Management.

For an overview of IAM users and why they are important for the security of your account, see AWS Security Credentials in the Amazon Web Services General Reference.

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.

Compliance validation for the AWS SDK for Kotlin

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.

The security and compliance of AWS services is assessed by third-party auditors as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others. AWS provides a frequently updated list of AWS services in scope of specific compliance programs at AWS services in Scope by Compliance Program.

Third-party audit reports are available for you to download using AWS Artifact. For more information, see Downloading Reports in AWS Artifact services.

For more information about AWS compliance programs, see AWS Compliance Programs.

Your compliance responsibility when using this AWS product or service to access an AWS service is determined by the sensitivity of your data, your organization's compliance objectives, and applicable laws and regulations. If your use of an AWS service is subject to compliance with standards such as HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- Security and Compliance Quick Start Guides - Deployment guides that discuss architectural considerations and provide steps for deploying security-focused and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper - A whitepaper that describe show companies can use AWS to create HIPAA-compliant applications.

- AWS Compliance Resources - A collection of workbooks and guides that might apply to your industry and location.
- AWS Config - A service that assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub - A comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience for this AWS Product or Service

The Amazon Web Services (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 Availability 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 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.

update-history-change	update-history-description	update-history-date
AWS SDK for Kotlin Developer Preview release (p. 138)	AWS SDK for Kotlin	December 2, 2021
AWS SDK for Kotlin Alpha release (p. 138)	Announcing new AWS SDK for Kotlin alpha release	August 30, 2021