Developer Guide for version 2.x

AWS SDK for Java 2.x

AWS SDK for Java 2.x: Developer Guide for version 2.x

Table of Contents

Developer Guide - AWS SDK for Java 2.x

The AWS SDK for Java provides a Java API for AWS services. Using the SDK, you can build Java applications that work with Amazon S3, Amazon EC2, DynamoDB, and more.

The AWS SDK for Java 2.x is a major rewrite of the version 1.x code base. It's built on top of Java 8+ and adds several frequently requested features. These include support for non-blocking I/O and the ability to plug in a different HTTP implementation at runtime.

We regularly add support for new services to the AWS SDK for Java. For a list of changes and features in a particular version, view the change log.

Get started with the SDK

If you're ready to get hands-on with the SDK, follow the *Get started tutorial* tutorial.

To set up your development environment, see *Setup*.

If you're currently using version 1.x of the SDK for Java, see Migrate to version 2 for specific guidance.

For information on making requests to Amazon S3, DynamoDB, Amazon EC2 and other AWS services, see Use the SDK for Java and Work with AWS services.

Develop mobile applications

If you're a mobile app developer, Amazon Web Services provides the AWS Amplify framework.

Maintenance and support for SDK major versions

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

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

Additional resources

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

- AWS SDK for Java 2.x API Reference
- Java developer blog
- Java development topic in AWS re:Post
- SDK source on GitHub
- AWS SDK Code Examples library
- @awsforjava (Twitter)

Contribute to the SDK

Developers can also contribute feedback through the following channels:

- Submit issues on GitHub:
  - Submit Developer Guide documentation issues
  - Submit SDK issues
- Join an informal chat about the SDK on the AWS SDK for Java 2.x gitter channel

Get started with the AWS SDK for Java 2.x

The AWS SDK for Java 2.x provides Java APIs for Amazon Web Services (AWS). Using the SDK, you can build Java applications that work with Amazon S3, Amazon EC2, DynamoDB, and more.

This tutorial shows you how to use Apache Maven to define dependencies for the SDK for Java 2.x and then write code that connects to Amazon S3 to upload a file.

Follow these steps to complete this tutorial:

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

Step 1: Set up for this tutorial

Before you begin this tutorial, you need the following:

- Permission to access Amazon S3
- A Java development environment that is configured to access AWS services using single sign-on to the AWS IAM Identity Center

Use the instructions in ??? to get set up for this tutorial. After you have configured your development environment with single sign-on access for the Java SDK and you have an active AWS access portal session, continue with Step 2 of this tutorial.

Step 2: Create the project

To create the project for this tutorial, you run a Maven command that prompts you for input on how to configure the project. After all input is entered and confirmed, Maven finishes building out the project by creating a `pom.xml` and creates stub Java files.

1. Open a terminal or command prompt window and navigate to a directory of your choice, for example, your `Desktop` or `Home` folder.

2. Enter the following command at the terminal and press `Enter`.

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

3. Enter the value listed in the second column for each prompt.

Prompt	Value to enter
`Define value for property 'service':`	`s3`
`Define value for property 'httpClient' :`	`apache-client`
`Define value for property 'nativeImage' :`	`false`
`Define value for property 'credentialProvider'`	`identity-center`
`Define value for property 'groupId':`	`org.example`
`Define value for property 'artifactId':`	`getstarted`
`Define value for property 'version' 1.0-SNAPSHOT:`	*<Enter>*
`Define value for property 'package' org.example:`	*<Enter>*

4. After the last value is entered, Maven lists the choices you made. Confirm by entering *Y* or re-enter values by entering *N*.

Maven creates the project folder named `getstarted` based on the `artifactId` value that you entered. Inside the `getstarted` folder, find a `README.md` file that you can review, a `pom.xml` file, and a `src` directory.

Maven builds the following directory tree.

```
getstarted
### README.md
### pom.xml
### src
    ### main
    #   ### java
    #   #   ### org
    #   #       ### example
    #   #           ### App.java
    #   #           ### DependencyFactory.java
    #   #           ### Handler.java
    #   ### resources
    #       ### simplelogger.properties
    ### test
        ### java
            ### org
                ### example
                    ### HandlerTest.java

10 directories, 7 files
```

The following shows the contents of the `pom.xml` project file.

pom.xml

The `dependencyManagement` section contains a dependency to the AWS SDK for Java 2.x and the `dependencies` section has a dependency for Amazon S3. The project uses Java 1.8 because of the `1.8` value in the `maven.compiler.source` and `maven.compiler.target` properties.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
```

```
    <artifactId>getstarted</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <maven.shade.plugin.version>3.2.1</maven.shade.plugin.version>
        <maven.compiler.plugin.version>3.6.1</maven.compiler.plugin.version>
        <exec-maven-plugin.version>1.6.0</exec-maven-plugin.version>
        <aws.java.sdk.version>2.20.43</aws.java.sdk.version> <-------- SDK version
 picked up from archetype version.
        <slf4j.version>1.7.28</slf4j.version>
        <junit5.version>5.8.1</junit5.version>
    </properties>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>${aws.java.sdk.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>  <-------- S3 dependency
            <exclusions>
                <exclusion>
                    <groupId>software.amazon.awssdk</groupId>
                    <artifactId>netty-nio-client</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>software.amazon.awssdk</groupId>
                    <artifactId>apache-client</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
```

```
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>sso</artifactId> <-------- Required for identity center
 authentication.
        </dependency>

        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>ssooidc</artifactId> <-------- Required for identity center
 authentication.
        </dependency>

        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>apache-client</artifactId> <-------- HTTP client specified.
            <exclusions>
                <exclusion>
                    <groupId>commons-logging</groupId>
                    <artifactId>commons-logging</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>${slf4j.version}</version>
        </dependency>

        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>${slf4j.version}</version>
        </dependency>

        <!-- Needed to adapt Apache Commons Logging used by Apache HTTP Client to Slf4j
 to avoid
        ClassNotFoundException: org.apache.commons.logging.impl.LogFactoryImpl during
 runtime -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>jcl-over-slf4j</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
```

```
        <!-- Test Dependencies -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>${junit5.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>${maven.compiler.plugin.version}</version>
            </plugin>
        </plugins>
    </build>

</project>
```

Step 3: Write the code

The following code shows the App class created by Maven. The main method is the entry point into the application, which creates an instance of the Handler class and then calls its sendRequest method.

App class

```
package org.example;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class App {
    private static final Logger logger = LoggerFactory.getLogger(App.class);

    public static void main(String... args) {
        logger.info("Application starts");

        Handler handler = new Handler();
        handler.sendRequest();
```

```
        logger.info("Application ends");
    }
}
```

The `DependencyFactory` class created by Maven contains the `s3Client` factory method that builds and returns an [`S3Client`] instance. The `S3Client` instance uses an instance of the Apache-based HTTP client. This is because you specified `apache-client` when Maven prompted you for which HTTP client to use.

The `DependencyFactory` is shown in the following code.

`DependencyFactory` class

```
package org.example;

import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;

/**
 * The module containing all dependencies required by the {@link Handler}.
 */
public class DependencyFactory {

    private DependencyFactory() {}

    /**
     * @return an instance of S3Client
     */
    public static S3Client s3Client() {
        return S3Client.builder()
                       .httpClientBuilder(ApacheHttpClient.builder())
                       .build();
    }
}
```

The `Handler` class contains the main logic of your program. When an instance of `Handler` is created in the App class, the `DependencyFactory` furnishes the `S3Client` service client. Your code uses the `S3Client` instance to call the Amazon S3 service.

Maven generates the following `Handler` class with a *TODO* comment. The next step in the tutorial replaces the *TODO* with code.

`Handler` class, Maven-generated

```
package org.example;

import software.amazon.awssdk.services.s3.S3Client;


public class Handler {
    private final S3Client s3Client;

    public Handler() {
        s3Client = DependencyFactory.s3Client();
    }

    public void sendRequest() {
        // TODO: invoking the api calls using s3Client.
    }
}
```

To fill in the logic, replace the entire contents of the `Handler` class with the following code. The `sendRequest` method is filled in and the necessary imports are added.

`Handler` class, implemented

The code first creates a new S3 bucket with the last part of the name generated using `System.currentTimeMillis()` in order to make the bucket name unique.

After creating the bucket in the `createBucket()` method, the program uploads an object using the [`putObject`] method of `S3Client`. The contents of the object is a simple string created with the `RequestBody.fromString` method.

Finally, the program deletes the object followed by the bucket in the `cleanUp` method.

```
package org.example;

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;


public class Handler {
    private final S3Client s3Client;

    public Handler() {
        s3Client = DependencyFactory.s3Client();
    }

    public void sendRequest() {
        String bucket = "bucket" + System.currentTimeMillis();
        String key = "key";

        createBucket(s3Client, bucket);

        System.out.println("Uploading object...");

        s3Client.putObject(PutObjectRequest.builder().bucket(bucket).key(key)
                        .build(),
                RequestBody.fromString("Testing with the {sdk-java}"));

        System.out.println("Upload complete");
        System.out.printf("%n");

        cleanUp(s3Client, bucket, key);

        System.out.println("Closing the connection to {S3}");
        s3Client.close();
        System.out.println("Connection closed");
        System.out.println("Exiting...");
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            s3Client.createBucket(CreateBucketRequest
                    .builder()
                    .bucket(bucketName)
                    .build());
            System.out.println("Creating bucket: " + bucketName);
            s3Client.waiter().waitUntilBucketExists(HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build());
            System.out.println(bucketName + " is ready.");
```

```
            System.out.printf("%n");
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void cleanUp(S3Client s3Client, String bucketName, String keyName) {
        System.out.println("Cleaning up...");
        try {
            System.out.println("Deleting object: " + keyName);
            DeleteObjectRequest deleteObjectRequest =
 DeleteObjectRequest.builder().bucket(bucketName).key(keyName).build();
            s3Client.deleteObject(deleteObjectRequest);
            System.out.println(keyName + " has been deleted.");
            System.out.println("Deleting bucket: " + bucketName);
            DeleteBucketRequest deleteBucketRequest =
 DeleteBucketRequest.builder().bucket(bucketName).build();
            s3Client.deleteBucket(deleteBucketRequest);
            System.out.println(bucketName + " has been deleted.");
            System.out.printf("%n");
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Cleanup complete");
        System.out.printf("%n");
    }
}
```

Step 4: Build and run the application

After the project is created and contains the complete `Handler` class, build and run the application.

1. Make sure that you have an active IAM Identity Center session. To do so, run the AWS Command Line Interface command `aws sts get-caller-identity` and check the response. If you don't have an active session, see this section for instructions.
2. Open a terminal or command prompt window and navigate to your project directory `getstarted`.
3. Use the following command to build your project:

```
mvn clean package
```

4. Use the following command to run the application.

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

To view the new bucket and object that the program creates, perform the following steps.

1. In `Handler.java`, comment out the line `cleanUp(s3Client, bucket, key)` in the `sendRequest` method and save the file.
2. Rebuild the project by running `mvn clean package`.
3. Rerun `mvn exec:java -Dexec.mainClass="org.example.App"` to upload the text object once more.
4. Sign in to the S3 console to view the new object in the newly created bucket.

After you view the file, delete the object, and then delete the bucket.

Success

If your Maven project built and ran without error, then congratulations! You have successfully built your first Java application using the SDK for Java 2.x.

Cleanup

To clean up the resources you created during this tutorial, do the following:

- If you haven't done so already, in the S3 console, delete any objects and any buckets created when you ran the application.
- Delete the project folder (`getstarted`).

Next steps

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

- Working with Amazon S3

- Working with other Amazon Web Services, such as DynamoDB, Amazon EC2, and various database services
- Use the SDK
- Security for the AWS SDK for Java

Set up the AWS SDK for Java 2.x

This section provides information about how to set up your development environment and projects to use the AWS SDK for Java 2.x.

Setup overview

To successfully develop applications that access AWS services using the AWS SDK for Java, the following conditions are required:

- The Java SDK must have access to credentials to authenticate requests on your behalf.
- The permissions of the IAM role configured for the SDK must allow access to the AWS services that your application requires. The permissions associated with the **PowerUserAccess** AWS managed policy are sufficient for most development needs.
- A development environment with the following elements:
  - Shared configuration files that are set up in at least one of the following ways:
    - The `config` file contains IAM Identity Center single sign-on settings so that the SDK can get AWS credentials.
    - The `credentials` file contains temporary credentials.
  - An installation of Java 8 or later.
  - A build automation tool such as Maven or Gradle.
  - A text editor to work with code.
  - (Optional, but recommended) An IDE (integrated development environment) such as IntelliJ IDEA, Eclipse, or NetBeans.

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

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

Important

The instructions in this setup section assume that you or organization uses IAM Identity Center. If your organization uses an external identity provider that works independently of IAM Identity Center, find out how you can get temporary credentials for the SDK for Java to use. Follow these instructions to add temporary credentials to the `~/.aws/credentials` file.

If your identity provider adds temporary credentials automatically to the `~/.aws/credentials` file, make sure that the profile name is `[default]` so that you do not need to provide a profile name to the SDK or AWS CLI.

Set up authentication

The Authentication and access topic in the AWS SDKs and Tools Reference Guide describes the different options to authenticate. We recommend that you follow the instructions to set up access to the IAM Identity Center so the SDK can acquire credentials. After following the instructions, your system is set up to allow the SDK to authenticate requests.

Setup for single sign-on access for the SDK

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

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

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

```
[default]
sso_session = my-sso
```

```
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

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

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

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

Sign in using the AWS CLI

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

```
aws sso login
```

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

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

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

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

> **Note**
>
> If you already have an active AWS access portal session and run `aws sso login`, you will not be required to provide credentials.

However, you will see a dialog that requests permission for `botocore` to access your information. `botocore` is the foundation for the AWS CLI .
Select **Allow** to authorize access to your information for the AWS CLI and SDK for Java.

Install Java and a build tool

Your development environment needs the following:

- Java 8 or later. The AWS SDK for Java 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 Adoptium.
- 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 https://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/.

Additional authentication options

For more options on authentication for the SDK, such as the use of profiles and environment variables, see the configuration chapter in the AWS SDKs and Tools Reference Guide.

Set up an Apache Maven project

You can use Apache Maven to set up and build AWS SDK for Java projects, or to build the SDK itself.

Prerequisites

To use the AWS SDK for Java with Maven, you need the following:

- Java *8.0 or later*. You can download the latest Java SE Development Kit software from http://www.oracle.com/technetwork/java/javase/downloads/. The AWS SDK for Java also works with OpenJDK and Amazon Corretto, a distribution of the Open Java Development Kit (OpenJDK). Download the latest OpenJDK version from https://openjdk.java.net/install/index.html. Download the latest Amazon Corretto 8 or Amazon Corretto 11 version from the Corretto page.

- *Apache Maven*. If you need to install Maven, go to http://maven.apache.org/ to download and install it.

Create a Maven project

To create a Maven project from the command line, run the following command from a terminal or command prompt window.

```
mvn -B archetype:generate \
 -DarchetypeGroupId=software.amazon.awssdk \
 -DarchetypeArtifactId=archetype-lambda -Dservice=s3 -Dregion=US_WEST_2 \
 -DarchetypeVersion=2.X.X \
 -DgroupId=com.example.myapp \
 -DartifactId=myapp
```

Note

Replace *com.example.myapp* with the full package namespace of your application. Also replace *myapp* with your project name. This becomes the name of the directory for your project.
To use the latest version of the archetype, replace *2.X.X* with the latest from Maven central.

This command creates a Maven project using the archetype templating toolkit. The archetype generates the scaffolding for an AWS Lambda function handler project . This project archetype is preconfigured to compile with Java SE 8 and includes a dependency to the version of the SDK for Java 2.x specified with `-DarchetypeVersion`.

For more information about creating and configuring Maven projects, see the Maven Getting Started Guide.

Configure the Java compiler for Maven

If you created your project using the AWS Lambda project archetype as described previously, the configuration of the Java compiler is already done for you.

To verify that this configuration is present, start by opening the `pom.xml` file from the project folder you created (for example, `myapp`) when you executed the previous command. Look on lines 11 and 12 to see the Java compiler version setting for this Maven project, and the required inclusion of the Maven compiler plugin on lines 71-75.

```
<project>
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>${maven.compiler.plugin.version}</version>
      </plugin>
    </plugins>
  </build>
</project>
```

If you create your project with a different archetype or by using another method, you must ensure that the Maven compiler plugin is part of the build and that its source and target properties are both set to **1.8** in the `pom.xml` file.

See the previous snippet for one way to configure these required settings.

Alternatively, you can configure the compiler configuration inline with the plugin declaration, as follows.

```
<project>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Declare the SDK as a dependency

To use the AWS SDK for Java in your project, you need to declare it as a dependency in your project's `pom.xml` file.

If you created your project using the project archetype as described previously, the latest version of the SDK is already configured as a dependency in your project.

The archetype generates a BOM (bill of materials) artifact dependency for the `software.amazon.awssdk` group id. With a BOM, you do not have to specify the maven version for individual artifact dependencies that share the same group id.

If you created your Maven project in a different way, configure the latest version of the SDK for your project by ensuring that the `pom.xml` file contains the following.

```
<project>
  <properties>
    <aws.java.sdk.version>2.X.X</aws.java.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.java.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

Note

Replace *2.X.X* in the `pom.xml` file with the [latest version of the AWS SDK for Java 2.x](#).

Set dependencies for SDK modules

Now that you have configured the SDK, you can add dependencies for one or more of the AWS SDK for Java modules to use in your project.

Although you can specify the version number for each component, you don't need to because you already declared the SDK version in the `dependencyManagement` section using the bill of materials artifact. To load a different version of a given module, specify a version number for its dependency.

If you created your project using the project archetype as described previously, your project is already configured with multiple dependencies. These include dependences for AWS Lambda function handlers and Amazon S3, as follows.

```
<project>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>software.amazon.awssdk</groupId>
                    <artifactId>netty-nio-client</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>software.amazon.awssdk</groupId>
                    <artifactId>apache-client</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>url-connection-client</artifactId>
        </dependency>

        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-lambda-java-core</artifactId>
            <version>${aws.lambda.java.version}</version>
        </dependency>
    </dependencies>
</project>
```

Note

In the `pom.xml` example above, the dependencies are from different `groupId`s. The `s3` dependency is from `software.amazon.awssdk`, whereas the `aws-lambda-java-core` dependency is from `com.amazonaws`. The BOM dependency management configuration affects artifacts for `software.amazon.awssdk`, so a version is needed for the `aws-lambda-java-core` artifact.

For the development of *Lambda function handlers* using the SDK for Java 2.x, `aws-lambda-java-core` is the correct dependency. However, if your application needs to manage Lambda resources, using operations such as `listFunctions`, `deleteFunction`, `invokeFunction`, and `createFunction`, your application requires the following dependency.

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

Note

The `s3` dependency excludes the the `netty-nio-client` and `apache-client` transitive dependencies. In place of either of those HTTP clients, the archetype includes the `url-connection-client` dependency, which helps reduce the startup latency for AWS Lambda functions.

Add the modules to your project for the AWS service and features you need for your project. The modules (dependencies) that are managed by the AWS SDK for Java BOM are listed on the Maven central repository.

Note

You can look at the `pom.xml` file from a code example to determine which dependencies you need for your project. For example, if you're interested in the dependencies for the DynamoDB service, see this example from the AWS Code Examples Repository on GitHub. (Look for the `pom.xml` file under /javav2/example_code/dynamodb.)

Build the entire SDK into your project

To optimize your application, we strongly recommend that you pull in only the components you need instead of the entire SDK. However, to build the entire AWS SDK for Java into your project, declare it in your pom.xml file, as follows.

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

Build your project

After you configure the pom.xml file, you can use Maven to build your project.

To build your Maven project from the command line, open a terminal or command prompt window, navigate to your project directory (for example, myapp), enter or paste the following command, then press Enter or Return.

```
mvn package
```

This creates a single .jar file (JAR) in the target directory (for example, myapp/target). This JAR contains all of the SDK modules you specified as dependencies in your pom.xml file.

Set up a Gradle project

You can use Gradle to set up and build AWS SDK for Java projects.

The initial steps in the following example come from Gradle's Getting Started guide for version 8.4. If you use a different version, your results may differ slightly.

To create a Java application with Gradle (command line)

1. Create a directory to hold your project. In this example, demo is the directory name.

2. Inside the demo directory, execute the `gradle init` command and supply the values highlighted in red as shown in the following command line output. For the walk through, we choose Kotlin as the build script DSL language, but a complete example for Groovy is also shown at the end of this topic.

```
> gradle init
Starting a Gradle Daemon (subsequent builds will be faster)

Select type of project to generate:
1: basic
2: application
3: library
4: Gradle plugin
Enter selection (default: basic) [1..4] 2

Select implementation language:
1: C++
2: Groovy
3: Java
4: Kotlin
5: Scala
6: Swift
Enter selection (default: Java) [1..6] 3

Generate multiple subprojects for application? (default: no) [yes, no] no
Select build script DSL:
1: Kotlin
2: Groovy
Enter selection (default: Kotlin) [1..2] <Enter>

Select test framework:
1: JUnit 4
2: TestNG
3: Spock
4: JUnit Jupiter
Enter selection (default: JUnit Jupiter) [1..4] 4

Project name (default: demo): <Enter>
Source package (default: demo): <Enter>
Enter target version of Java (min. 7) (default: 11): <Enter>
Generate build using new APIs and behavior (some features may change in the next
 minor release)? (default: no) [yes, no] <Enter>
```

```
> Task :init
To learn more about Gradle by exploring our Samples at https://docs.gradle.org/8.4/
samples/sample_building_java_applications.html

BUILD SUCCESSFUL in 3m 43s
2 actionable tasks: 2 executed
```

3. After the `init` task completes, the `demo` directory contains the following tree structure. We take a closer look at the main build file, `build.gradle.kts` (highlighted in red), in the next section.

```
### app
#   ### build.gradle.kts
#   ### src
#       ### main
#       #   ### java
#       #   #   ### demo
#       #   #       ### App.java
#       #   ### resources
#       ### test
#           ### java
#           #   ### demo
#           #       ### AppTest.java
#           ### resources
### gradle
#   ### wrapper
#       ### gradle-wrapper.jar
#       ### gradle-wrapper.properties
### gradlew
### gradlew.bat
### settings.gradle.kts
```

The `build.gradle.kts` file contains the following scaffolded content.

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

```
 */

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

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

dependencies {
    // Use JUnit Jupiter for testing.
    testImplementation("org.junit.jupiter:junit-jupiter:5.9.3")

    testRuntimeOnly("org.junit.platform:junit-platform-launcher")

    // This dependency is used by the application.
    implementation("com.google.guava:guava:32.1.1-jre")
}

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

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

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

4. Use the scaffolded Gradle build file as the basis for your AWS project.

a. To manage SDK dependencies for your Gradle project, add the Maven bill of materials (BOM) for the AWS SDK for Java 2.x to the `dependencies` section of the `build.gradle.kts` file.

```
...
dependencies {
    implementation(platform("software.amazon.awssdk:bom:2.21.1"))
    // With the bom declared, you specify individual SDK dependencies without a
 version.
 ...
}
...
```

Note

In this example build file, replace *2.21.1* with the latest version of the SDK for Java 2.x. Find the latest version available in [Maven central repository](https://central.sonatype.com/artifact/software.amazon.awssdk/bom/versions).

b. Specify the SDK modules your application needs in the `dependencies` section. As an example, the following adds a dependency on Amazon Simple Storage Service.

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

Gradle automatically resolves the correct version of declared dependencies by using the information from the BOM.

The following examples show complete Gradle build files in both the Kotlin and Groovy DSLs. The build file contains dependencies for Amazon S3, authentication, logging, and testing. The source and target version of Java is version 11.

Kotlin DSL (build.gradle.kts)

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

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

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

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

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

application {
    // Define the main class for the application.
```

```
    mainClass.set("demo.App")
}

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

Groovy DSL (build.gradle)

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

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

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

dependencies {
    implementation platform('software.amazon.awssdk:bom:2.21.1')
    implementation 'software.amazon.awssdk:s3'
    implementation 'software.amazon.awssdk:sso'
    implementation 'software.amazon.awssdk:ssooidc'
    implementation platform('org.apache.logging.log4j:log4j-bom:2.20.0')
    implementation 'org.apache.logging.log4j:log4j-slf4j2-impl'
    implementation 'org.apache.logging.log4j:log4j-1.2-api'
    testImplementation platform('org.junit:junit-bom:5.10.0')
    testImplementation 'org.junit.jupiter:junit-jupiter'
}
```

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

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

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

For next steps, see the Getting Started guide on the Gradle website for instructions on how to build and run a Gradle application.

Set up a GraalVM Native Image project for the AWS SDK for Java

With versions 2.16.1 and later, the AWS SDK for Java provides out-of-the-box support for GraalVM Native Image applications. Use the `archetype-app-quickstart` Maven archetype to set up a project with built-in native image support.

Prerequisites

- Complete the steps in Setting up the AWS SDK for Java 2.x.
- Install GraalVM Native Image.

Create a project using the archetype

To create a Maven project with built-in native image support, in a terminal or command prompt window, use the following command.

> **Note**
>
> Replace `com.example.mynativeimageapp` with the full package namespace of your application. Also replace `mynativeimageapp` with your project name. This becomes the name of the directory for your project.

```
mvn archetype:generate \
    -DarchetypeGroupId=software.amazon.awssdk \
    -DarchetypeArtifactId=archetype-app-quickstart \
    -DarchetypeVersion=2.16.1 \
    -DnativeImage=true \
    -DhttpClient=apache-client \
    -Dservice=s3 \
    -DgroupId=com.example.mynativeimageapp \
    -DartifactId=mynativeimageapp \
    -DinteractiveMode=false
```

This command creates a Maven project configured with dependencies for the AWS SDK for Java, Amazon S3, and the `ApacheHttpClient` HTTP client. It also includes a dependency for the GraalVM Native Image Maven plugin, so that you can build native images using Maven.

To include dependencies for a different Amazon Web Services, set the value of the `-Dservice` parameter to the artifact ID of that service. Examples include `dynamodb`, `comprehend`, and `pinpoint`. For a complete list of artifact IDs, see the list of managed dependencies for software.amazon.awssdk on Maven Central.

To use an asynchronous HTTP client, set the `-DhttpClient` parameter to `netty-nio-client`. To use `UrlConnectionHttpClient` as the synchronous HTTP client instead of `apache-client`, set the `-DhttpClient` parameter to `url-connection-client`.

Build a native image

After you create the project, run the following command from your project directory, for example, `mynativeimageapp`:

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

This creates a native image application in the `target` directory, for example, `target/mynativeimageapp`.

Use the AWS SDK for Java 2.x

After completing the steps in Setting up the SDK, you are ready to make requests to AWS services such as Amazon S3, DynamoDB, IAM, Amazon EC2, and more.

Work with service clients

Create a service client

To make a request to an AWS service, you must first instantiate a service client for that service by using the static factory method, `builder()`. The `builder()` method returns a `builder` object that allows you to customize the service client. The fluent setter methods return the `builder` object, so that you can chain the method calls for convenience and for more readable code. After you configure the properties you want, call the `build()` method to create the client.

As an example, the following code snippet instantiates an `Ec2Client` object as a service client for Amazon EC2.

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

Note

Service clients in the SDK are thread-safe. For best performance, treat them as long-lived objects. Each client has its own connection pool resource that is released when the client is garbage collected.
A service client object is immutable, so you must create a new client for each service to which you make requests, or if you want to use a different configuration for making requests to the same service.
Specifying the `Region` in the service client builder is not required for all AWS services; however, it is a best practice to set the Region for the API calls you make in your applications. See AWS region selection for more information.

Default client configuration

The client builders have another factory method named `create()`. This 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 to `create` fails. See Using credentials and Region selection for more information about how the SDK determines the credentials and Region to use.

For example, the following code snippet instantiates a `DynamoDbClient` object as a service client for Amazon DynamoDB:

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

Configure service clients

To customize the configuration of a service client, use the setters on the `builder()` factory method. For convenience and to create more readable code, chain the methods to set multiple configuration options.

The following example shows an `S3Client` that is configured with several custom settings.

```
ClientOverrideConfiguration clientOverrideConfiguration =
    ClientOverrideConfiguration.builder()
                     .apiCallAttemptTimeout(Duration.ofSeconds(1))
                     .retryPolicy(RetryPolicy.builder().numRetries(10).build())
                     .addMetricPublisher(CloudWatchMetricPublisher.create())
                     .build();

Region region = Region.US_WEST_2;
S3Client s3Client = S3Client.builder()
                     .region(region)

 .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                     .overrideConfiguration(clientOverrideConfiguration)
                     .httpClientBuilder(ApacheHttpClient.builder()

 .proxyConfiguration(proxyConfig.build(ProxyConfiguration.builder()))
                         .build())
                     .build();
```

Make requests

Use the service client to make requests to the corresponding AWS service.

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

```
// Create the request by using the fluid setter methods of the request builder.
RunInstancesRequest runInstancesRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

// Use the configured request with the service client.
RunInstancesResponse response = ec2Client.runInstances(runInstancesRequest);
```

Rather than create a request and pass in the instance, the SDK provides builders that you can use to create a request. With a builder you can use a Java lambda expressions to create the request 'in-line'.

The following example rewrites the previous example by using the version of the `runInstances` method that uses a builder to create the request.

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

Handle responses

The SDK returns a response object for most service operations. Your code can process the information in the response object according to your needs.

For example, the following code snippet prints out the first instance id returned with the `RunInstancesResponse` object from the previous request.

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

Not all operations return a response object with service-specific data, however. In these situations, you can query the HTTP response status to learn if the operation was successful.

For example, the code in the following snippet checks the HTTP response to see if the `DeleteContactList` operation of Amazon Simple Email Service was successful.

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

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

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

Close the service client

As a best practice, you should use a service clients for multiple API service calls during the life of an application. However, if you need a service client for a one-time use or no longer need the service client, close it.

Call the `close()` method when the service client is no longer needed to free up resources.

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

If you need a service client for one-time use, you can instantiate the service client as a resource in a `try`-with-resources statement. Service clients implement the `Autoclosable` interface, so the JDK automatically calls the `close()` method at the end of the statement.

The following example demonstrates how to use a service client for a one-off call. The `StsClient` that calls the AWS Security Token Service is closed after it returns the account ID.

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

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

Handle exceptions

The SDK uses runtime (or unchecked) exceptions, providing you fine-grained control over error handling and ensuring that exception handling will scale with your application.

An `SdkServiceException`, or one of its sub-classes, is the most common form of exception the SDK will throw. These exceptions represent responses from the AWS service. You can also handle an `SdkClientException`, 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 when you upload a file to Amazon S3. The example code catches both client and server exceptions, logs the details, and exists the application.

```
Region region = Region.US_WEST_2;
s3Client = S3Client.builder()
        .region(region)
        .build();

try {

    PutObjectRequest putObjectRequest = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .build();

    s3Client.putObject(putObjectRequest, RequestBody.fromString("SDK for Java test"));

} catch (S3Exception se) {
    System.err.println("Service exception thrown.");
    System.err.println(se.awsErrorDetails().errorMessage());
} catch (SdkClientExeption ce){
   System.err.println("Client exception thrown.");
```

```
    System.err.println(ce.getMessage());
} finally {
    System.exit(1);
}
```

See Handling exceptions for more information.

Use waiters

Some requests take time to process, such as creating a new table in DynamoDB or creating a new Amazon S3 bucket. To ensure the resource is ready before your code continues to run, use a *Waiter*.

For example, this code snippet creates a new table ("myTable") in DynamoDB, waits for the table to be in an `ACTIVE` status, and then prints out the response:

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

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

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

See Using waiters for more information.

HTTP clients

You can change the default configuration for HTTP clients in applications you build with the AWS SDK for Java. For information on how to configure HTTP clients and settings, see HTTP configuration.

Timeouts

You can configure timeouts for each of your service clients using the `apiCallTimeout` and the `apiCallAttemptTimeout` setters of the `ClientOverrideConfiguration.Builder`. The `apiCallTimeout` setting is the amount of time to allow the client to complete the execution of an API call. The `apiCallAttemptTimeout` setting is the amount of time to wait for each HTTP request (retry) to complete before giving up.

The following example sets both timeouts for an S3 client.

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

You can also set timeouts at the request level by configuring an `AwsRequestOverrideConfiguration` and providing it to the request object with the `overrideConfiguration` method.

The following example uses the same timeout settings but at the request level for an S3 `PutObject` operation.

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

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

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

Execution interceptors

You can write code that intercepts the execution of your API requests and responses at different parts of the request/response lifecycle. This enables you to publish metrics, modify a request in-flight, debug request processing, view exceptions, and more. For more information, see the `ExecutionInterceptor` interface in the AWS SDK for Java API Reference.

Additional information

- For complete examples of the code snippets above, see Working with Amazon DynamoDB, Working with Amazon EC2, and Working with Amazon S3.

Provide temporary credentials to the SDK

Before making a request to Amazon Web Services using the AWS SDK for Java 2.x, the SDK cryptographically signs temporary credentials issued by AWS. To access temporary credentials, the SDK retrieves configuration values by checking several locations.

This topic discusses several ways that you enable the SDK to access temporary credentials.

Topics

Configure access to temporary credentials

For increased security, AWS recommends that you configure the SDK for Java to use temporary credentials instead of long-lived credentials. Temporary credentials consist of access keys (access key id and secret access key) and a session token. We recommend that you configure the SDK to automatically get temporary credentials, since the token refresh process is automatic. You can, however, provide the SDK with temporary credentials directly.

IAM Identity Center configuration

When you configure the SDK to use IAM Identity Center single sign-on access as described in ??? in this guide, the SDK automatically uses temporary credentials.

The SDK uses the IAM Identity Center access token to gain access to the IAM role that is configured with the `sso_role_name` setting in your `config` file. The SDK assumes this IAM role and retrieves temporary credentials to use for AWS service requests.

For more details about how the SDK gets temporary credentials from the configuration, see the Understanding IAM Identity Center authentication section of the AWS SDKs and Tools Reference Guide.

Retrieve from AWS access portal

As an alternative to IAM Identity Center single sign-on configuration, you can copy and use temporary credentials available in the AWS access portal. You can use the temporary credentials in a profile or use them as values for system properties and environment variables.

Set up a local credentials file for temporary credentials

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

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

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

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

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

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

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

7. Save the `credentials` file.

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

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

Default credentials provider chain

The default credentials provider chain is implemented by the DefaultCredentialsProvider class. It sequentially checks each place where you can set the default configuration for supplying temporary credentials, and then selects the first one you set.

To use the default credentials provider chain to supply temporary credentials, create a service client builder but don't specify a credentials provider. The following code snippet creates a `DynamoDbClient` that uses the default credentials provider chain to locate and retrieve default configuration settings.

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

Credential settings retrieval order

The default credentials provider chain of the SDK for Java 2.x searches for configuration in your environment using a predefined sequence.

1. Java system properties
   - The SDK uses the SystemPropertyCredentialsProvider class to load temporary credentials from the `aws.accessKeyId`, `aws.secretAccessKey`, and `aws.sessionToken` Java system properties.

   > **Note**
   >
   > For information on how to set Java system properties, see the System Properties tutorial on the official *Java Tutorials* website.

2. Environment variables

- The SDK uses the EnvironmentVariableCredentialsProvider class to load temporary credentials from the `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, and `AWS_SESSION_TOKEN` environment variables.

3. Web identity token from AWS Security Token Service
   - The SDK uses the WebIdentityTokenFileCredentialsProvider class to load temporary credentials from Java system properties or environment variables.
4. The shared `credentials` and `config` files
   - The SDK uses the ProfileCredentialsProvider to load IAM Identity Center single sign-on settings or temporary credentials from the `[default]` profile in the shared `credentials` and `config` files.

     The AWS SDKs and Tools Reference Guide has detailed information about how the SDK for Java works with the IAM Identity Center single sign-on token to get temporary credentials that the SDK uses to call AWS services.

     > **Note**
     >
     > The `credentials` and `config` files are shared by various AWS SDKs and Tools. For more information, see The .aws/credentials and .aws/config files in the AWS SDKs and Tools Reference Guide.
5. Amazon ECS container credentials
   - The SDK uses the ContainerCredentialsProvider class to load temporary credentials from the `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` system environment variable.
6. Amazon EC2 instance IAM role-provided credentials
   - The SDK uses the InstanceProfileCredentialsProvider class to load temporary credentials from the Amazon EC2 metadata service.

Use a specific credentials provider or provider chain

As an alternative to the default credentials provider chain, you can specify which credentials provider the SDK should use. When you supply a specific credentials provider, the SDK skips the process of checking various locations, which slightly reduces the time to create a service client.

For example, if you set your default configuration using environment variables, supply an EnvironmentVariableCredentialsProvider object to the `credentialsProvider` method on the service client builder, as in the following code snippet.

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

For a complete list of credential providers and provider chains, see **All Known Implementing Classes** in AwsCredentialsProvider.

Note

You can use your own credentials provider or provider chains by implementing the `AwsCredentialsProvider` interface.

Use profiles

Using the shared `config` and `credentials` file, you can set up several profiles. This enables your application to use multiple sets of credentials configuration. The `[default]` profile was mentioned previously. The SDK uses the ProfileCredentialsProvider class to load settings from profiles defined in the shared `credentials` file.

The following code snippet demonstrates how to build a service client that uses the settings defined as part of the profile named `my_profile`.

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

Set a different profile as the default

To set a profile other than the `[default]` profile as the default for your application, set the `AWS_PROFILE` environment variable to the name of your custom profile.

To set this variable on Linux, macOS, or Unix, use `export`:

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

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

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

Alternatively, set the `aws.profile` Java system property to the name of the profile.

Reload profile credentials

You can configure any credentials provider that has a `profileFile()` method on its builder to reload profile credentials. These credentials profile classes are: `ProfileCredentialsProvider`, `DefaultCredentialsProvider`, `InstanceProfileCredentialsProvider`, and `ProfileTokenProvider`.

> **Note**
>
> Profile credential reloading works only with the following settings in the profile file : `aws_access_key_id`, `aws_secret_access_key`, and `aws_session_token`. Settings such as `region`, `sso_session`, `sso_account_id`, and `source_profile` are ignored.

To configure a supported credentials provider to reload profile settings, provide an instance of [`ProfileFileSupplier`] to the `profileFile()` builder method. The following code example demonstrates a `ProfileCredentialsProvider` that reloads credential settings from the `[default]` profile.

```
ProfileCredentialsProvider provider = ProfileCredentialsProvider
    .builder()
    .profileFile(ProfileFileSupplier.defaultSupplier())
    .build();

// Set up a service client with the provider instance.
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .credentialsProvider(provider)
                    .build();
```

```
/*
    Before dynamoDbClient makes a request, it reloads the credentials settings
    by calling provider.resolveCredentials().
*/
```

When `ProfileCredentialsProvider.resolveCredentials()` is called, the SDK for Java reloads the settings. `ProfileFileSupplier.defaultSupplier()` is one of several convenience implementations of `ProfileFileSupplier` provided by the SDK. If your use case requires, you can provide your own implementation.

The following example shows the use of the `ProfileFileSupplier.reloadWhenModified()` convenience method. `reloadWhenModified()` takes a `Path` parameter, which gives you flexibility in designating the source file for the configuration rather than the standard `~/.aws/credentials` (or `config`) location.

The settings will be reloaded when `resolveCredentials()` is called only if SDK determines the file's content has been modified.

```
Path credentialsFilePath = ...

ProfileCredentialsProvider provider = ProfileCredentialsProvider
    .builder()
    .profileFile(ProfileFileSupplier.reloadWhenModified(credentialsFilePath,
 ProfileFile.Type.CREDENTIALS))
    .profileName("my-profile")
    .build();
/*
    A service client configured with the provider instance calls
 provider.resolveCredential()
    before each request.
*/
```

The `ProfileFileSupplier.aggregate()` method merges the contents of multiple configuration files. You decide whether a file is reloaded per call to `resolveCredentials()` or a file's settings are fixed at the time it was first read.

The following example shows a `DefaultCredentialsProvider` that merges the settings of two files that contain profile settings. The SDK reloads the settings in the file pointed to by the `credentialsFilePath` variable each time `resolveCredentials()` is called and the settings have changed. The settings from the `profileFile` object remain the same.

```
Path credentialsFilePath = ...;
ProfileFile profileFile = ...;

DefaultCredentialsProvider provider = DefaultCredentialsProvider
        .builder()
        .profileFile(ProfileFileSupplier.aggregate(
                ProfileFileSupplier.reloadWhenModified(credentialsFilePath,
 ProfileFile.Type.CREDENTIALS),
                ProfileFileSupplier.fixedProfileFile(profileFile)))
        .profileName("my-profile")
        .build();
/*
    A service client configured with the provider instance calls
 provider.resolveCredential()
    before each request.
*/
```

Load temporary credentials from an external process

Warning

The following describes a method of sourcing temporary credentials from an external process. This can potentially be dangerous, so proceed with caution. Other credential providers should be preferred if at all possible. If using this option, you should make sure that the `config` file is as locked down as possible using security best practices for your operating system.
Make sure that your custom credentials tool does not write any secret information to `StdErr`. SDKs and AWS CLI can capture and log such information, potentially exposing it to unauthorized users.

With the SDK for Java 2.x, you can acquire temporary credentials from an external process for custom use cases. There are two ways to configure this functionality.

Use the `credential_process` setting

If you have a method that provides temporary credentials, you can integrate it by adding the `credential_process` setting as part of a profile definition in the `config` file. The value you specify must use the full path to the command file. If the file path contains any spaces, you must surround it with quotation marks.

The SDK calls the command exactly as given and then reads JSON data from `stdout`.

The following examples show the use of this setting for file paths without spaces and file paths with spaces.

Linux/macOS

No spaces in file path

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

Spaces in file path

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

Windows

No spaces in file path

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

Spaces in file path

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

The following code snippet demonstrates how to build a service client that uses the temporary credentials defined as part of the profile named `process-credential-profile`.

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

```
        .build();
```

For detailed information about using an external process as a source of temporary credentials, refer to the process credentials section in the AWS SDKs and Tools Reference Guide.

Use the `ProcessCredentialsProvider`

As an alternative to using settings in the `config` file, you can use the SDK's `ProcessCredentialsProvider` to load temporary credentials using Java.

The following examples show various versions of how to specify an external process using the `ProcessCredentialsProvider` and configuring a service client that uses the temporary credentials.

Linux/macOS

No spaces in file path

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("/path/to/credentials.sh optional_param1 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
                      .region(Region.US_WEST_2)
                      .credentialsProvider(credentials)
                      .build();
```

Spaces in file path

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("/path\\ with\\ spaces\\ to/credentials.sh optional_param1
 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
                      .region(Region.US_WEST_2)
                      .credentialsProvider(credentials)
```

```
                       .build();
```

Windows

No spaces in file path

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("C:\\Path\\To\\credentials.exe optional_param1 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
                      .region(Region.US_WEST_2)
                      .credentialsProvider(credentials)
                      .build();
```

Spaces in file path

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("\"C:\\Path\\With Spaces To\\credentials.exe\" optional_param1
 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
                      .region(Region.US_WEST_2)
                      .credentialsProvider(credentials)
                      .build();
```

Supply temporary credentials in code

If the default credential chain or a specific or custom provider or provider chain doesn't work for your application, you can supply temporary credentials directly in code. These can be IAM role credentials as described above or temporary credentials retrieved from AWS Security Token Service (AWS STS). If you retrieved temporary credentials using AWS STS, provide them to an AWS service client as shown in the following code example.

1. Assume a role by calling `StsClient.assumeRole()`.

2. Create a StaticCredentialsProvider object and supply it with the `AwsSessionCredentials` object.
3. Configure the service client builder with the `StaticCredentialsProvider` and build the client.

The following example creates an Amazon S3 service client using temporary credentials returned by AWS STS for an IAM assumed role.

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

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

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

            // List all buckets in the account associated with the assumed role
            // by using the temporary credentials retrieved by invoking
stsClient.assumeRole().
            StaticCredentialsProvider staticCredentialsProvider =
StaticCredentialsProvider.create(
                    AwsSessionCredentials.create(key, secKey, secToken));
```

```
            try (S3Client s3 = S3Client.builder()
                    .credentialsProvider(staticCredentialsProvider)
                    .build()) {
                List<Bucket> buckets = s3.listBuckets().buckets();
                for (Bucket bucket : buckets) {
                    System.out.println("bucket name: " + bucket.name());
                }
            }
        } catch (StsException | S3Exception e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }
```

Permission set

The following permission set defined in AWS IAM Identity Center allows the identity (user) to perform the following two operations

1. The `GetObject` operation of the Amazon Simple Storage Service.
2. The `AssumeRole` operation of the AWS Security Token Service.

Without assuming the role, the `s3.listBuckets()` method shown in the example would fail.

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

Assumed role

Assumed role permissions policy

The following permissions policy is attached to the role that is assume in the previous example. This permissions polilcy permits the ability to list all buckets in the same account as the role.

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

Assumed role trust policy

The following trust policy is attached to the role that is assume in the previous example. The policy allows the role to be assumed by identities (users) in two accounts.

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

Read IAM role credentials on Amazon EC2

You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

This topic provides information on how to set up your Java application to run on an EC2 instance and enable the SDK for Java to acquire IAM role credentials.

Acquire IAM role credentials from the environment

If your application creates an AWS service client by using the `create` method (or `builder().build()` methods), the SDK for Java uses the *default credentials provider chain*. The default credentials provider chain searches the execution environment for configuration elements that the SDK can trade for temporary credentials. The the section called "Default credentials provider chain" section describes the full search process.

The final step in the default provider chain is available only when your application runs on an Amazon EC2 instance. In this step, the SDK uses an `InstanceProfileCredentialsProvider` to read the IAM role defined in the EC2 instance profile. The SDK then acquires temporary credentials for that IAM role.

Although these credentials are temporary and would eventually expire, an `InstanceProfileCredentialsProvider` periodically refreshes them for you so that they continue to allow access to AWS.

Acquire IAM role credentials programmatically

As an alternative to the default credentials provider chain that eventually uses an `InstanceProfileCredentialsProvider` on EC2, you can configure a service client explicitly with an `InstanceProfileCredentialsProvider`. This approach is shown in the following snippet.

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

Securely acquire IAM role credentials

By default, EC2 instances run IMDS (Instance Metadata Service) that allows the SDK's `InstanceProfileCredentialsProvider` to access information such as the IAM role that has been configured. EC2 instances run two versions of IMDS by default:

- Instance Metadata Service Version 1 (IMDSv1) – a request/response method
- Instance Metadata Service Version 2 (IMDSv2) – a session-oriented method

IMDSv2 is a more secure approach than IMDSv1.

By default, the Java SDK first tries IMDSv2 to get the IAM role, but if that fails, it tries IMDSv1. However, since IMDSv1 is less secure, AWS recommends the use of IMDSv2 only and to disable the SDK from trying IMDSv1.

To use the more secure approach, disable the SDK from using IMDSv1 by providing one of the following settings with a value of `true`.

- Environment variable: `AWS_EC2_METADATA_V1_DISABLED`
- JVM system property: aws.`disableEc2MetadataV1`
- Shared config file setting: `ec2_metadata_v1_disabled`

With one of these settings set to `true`, the SDK does not load IMDS role credentials by using IMDSv1 if the initial IMDSv2 call fails.

Use AWS Regions

AWS Regions enable service clients to access AWS services that physically resides in a specific geographic area.

Explicitly configure an AWS Region

To explicitly set a Region, we recommend that you use the constants defined in the Region class. This is an enumeration of all publicly available regions.

To create a client with an enumerated Region from the class, use the client builder's `region` method.

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

If the Region you want to use isn't one of the enumerations in the `Region` class, you can create a new Region by using the static `of` method. This method allows you access to new Regions without upgrading the SDK.

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

Note

After you build a client with the builder, it's *immutable* and the AWS Region *cannot be changed*. If you need to work with multiple AWS Regions for the same service, you should create multiple clients—one per Region.

Let the SDK automatically determine the Region from the environment

When your code runs on Amazon EC2 or AWS Lambda, you might want to configure clients to use the same AWS 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 AWS Regions for lower latency or redundancy.

To use the default credential/region provider chain to determine the Region from the environment, use the client builder's `create` method.

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

If you don't explicitly set an AWS Region by using the `region` method, the SDK consults the default region provider chain to determine the Region to use.

Understand the default region provider chain

The SDK takes the following steps to look for an AWS Region :

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

   > **Note**
   >
   > The Lambda container sets this environment variable.

3. The SDK checks the AWS shared configuration file and shared credentials file (usually located at `~/.aws/config` and `~/.aws/credentials`). If the `region` property is present, the SDK uses it.
   - If the SDK finds the `region` property in both files for the same profile (including the `default` profile), the SDK uses the value in the shared credentials file.
   - 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 specify the profile that the SDK loads.
4. The SDK attempts to use the Amazon EC2 instance metadata service (IMDS) to determine the Region of the currently running Amazon EC2 instance.
   - For greater security, you should disable the SDK from attempting to use version 1 of IMDS. You use the same setting to disable version 1 that are described in the the section called "Securely" section.
5. If the SDK still hasn't found a Region by this point, client creation fails with an exception.

When developing AWS applications, a common approach is to use the *shared configuration file* (described in Credential retrieval order) to set the Region for local development, and rely on the default region provider chain to determine the Region when the application runs on AWS infrastructure. This greatly simplifies client creation and keeps your application portable.

Check for service availability in a Region

To see if a particular AWS service is available in a Region, use the `serviceMetadata` and `region` method on the service client.

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

See the Region class documentation for the AWS Regions you can specify, and use the endpoint prefix of the service to query.

Choose a specific endpoint

In certain situations—such as to test preview features of a service before the features graduate to general availability—you may need to specify a specific endpoint in a Region. In these situations, service clients can be configured by calling the `endpointOverride` method.

For example, to configure an Amazon EC2 client to use the Europe (Ireland) Region with a specific endpoint, use the following code.

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

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

Reduce SDK startup time for AWS Lambda

One of the goals of the AWS SDK for Java 2.x is to reduce the startup latency for AWS Lambda functions. The SDK contains changes that reduce startup time, which are discussed at the end of this topic.

First, this topic focuses on changes that you can make to reduce cold start times. These include making changes in your code structure and in the configuration of service clients.

Use an AWS CRT-based HTTP client

For working with AWS Lambda, we recommend the `AwsCrtHttpClient` for synchronous scenarios and the `AwsCrtAsyncHttpClient` for asynchronous scenarios.

The the section called “Configure AWS CRT-based HTTP clients” topic in this guide describes the benefits of using the HTTP clients, how to add the dependency, and how configure their use by service clients.

Remove unused HTTP client dependencies

Along with the explicit use of an AWS CRT-based client, you can remove other HTTP clients that the SDK brings in by default. Lambda startup time is reduced when fewer libraries need to be loaded, so you should remove any unused artifacts that the JVM needs to load.

The following snippet of a Maven `pom.xml` file shows the exclusion of the Apache-based HTTP client and the Netty-based HTTP client. (These clients aren't needed when you use an AWS CRT-based client.) This example excludes the HTTP client artifacts from the S3 client dependency and adds the `aws-crt-client` artifact to allow access to the AWS CRT-based HTTP clients.

```
<project>
    <properties>
        <aws.java.sdk.version>2.25.51</aws.java.sdk.version>
    <properties>
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>${aws.java.sdk.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>aws-crt-client</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>software.amazon.awssdk</groupId>
                    <artifactId>netty-nio-client</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>software.amazon.awssdk</groupId>
                    <artifactId>apache-client</artifactId>
```

```
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>
</project>
```

Note

Add the `<exclusions>` element to all service client dependencies in your `pom.xml` file.

Configure service clients to shortcut lookups

Specify a region

When you create a service client, call the `region` method on the service client builder. This shortcuts the SDK's default Region lookup process that checks several places for the AWS Region information.

To keep the Lambda code independent of the region, use the following code inside the `region` method. This code accesses the `AWS_REGION` environment variable set by the Lambda container.

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

Use the `EnvironmentVariableCredentialProvider`

Much like the default lookup behavior for the Region information, the SDK looks in several places for credentials. By specifying the `EnvironmentVariableCredentialProvider` when you build a service client, you save time in the SDK's lookup process.

Note

Using this credentials provider enables the code to be used in Lambda functions, but might not work on Amazon EC2 or other systems.

The following code snippet shows an S3 service client appropriately configured for use in a Lambda environment.

```
S3Client s3Client = S3Client.builder()

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

Initialize the SDK client outside of the Lambda function handler

We recommend initializing an SDK client outside of the Lambda handler method. This way, if the execution context is reused, the initialization of the service client can be skipped. By reusing the client instance and its connections, subsequent invocations of the handler method occur more quickly.

In the following example, the `S3Client` instance is initialized in the constructor using a static factory method. If the container that is managed by the Lambda environment is reused, the initialized `S3Client` instance is reused.

```
public class App implements RequestHandler<Object, Object> {
    private final S3Client s3Client;

    public App() {
        s3Client = DependencyFactory.s3Client();
    }

    @Override
    public Object handle Request(final Object input, final Context context) {
         ListBucketResponse response = s3Client.listBuckets();
         // Process the response.
    }
}
```

Minimize dependency injection

Dependency injection (DI) frameworks might take additional time to complete the setup process. They might also require additional dependencies, which take time to load.

If a DI framework is needed, we recommend using lightweight DI frameworks such as Dagger.

Use a Maven Archetype targeting AWS Lambda

The AWS Java SDK team has developed a Maven Archetype template to bootstrap a Lambda project with minimal startup time. You can build out a Maven project from the archetype and know that the dependencies are configured suitably for the Lambda environment.

To learn more about the archetype and work through an example deployment, see this blog post.

Consider Lambda SnapStart for Java

If your runtime requirements are compatible, AWS offers Lambda SnapStart for Java. Lambda SnapStart is an infrastructure-based solution that improves startup performance for Java functions. When you publish a new version of a function, Lambda SnapStart initializes it and takes an immutable, encrypted snapshot of the memory and disk state. SnapStart then caches the snapshot for reuse.

Version 2.x changes that affect startup time

In addition to changes that you make to your code, version 2.x of the SDK for Java includes three primary changes that reduce startup time:

- Use of jackson-jr, which is a serialization library that improves initialization time
- Use of the java.time libraries for date and time objects, which is part of the JDK
- Use of Slf4j for a logging facade

Additional resources

The AWS Lambda Developer Guide contains a section on best practices for developing Lambda functions that is not Java specific.

For an example of building a cloud-native application in Java that uses AWS Lambda, see this workshop content. The workshop discussion performance optimization and other best practices.

You can consider using static images that are compiled ahead of time to reduce startup latency. For example, you can use the SDK for Java 2.x and Maven to build a GraalVM native image.

HTTP clients

You can change the HTTP client to use for your service client as well as change the default configuration for HTTP clients with the AWS SDK for Java 2.x. This section discusses HTTP clients and settings for the SDK.

HTTP clients available in the SDK for Java

Synchronous clients

Synchronous HTTP clients in the SDK for Java implement the SdkHttpClient interface. A synchronous service client, such as the `S3Client` or the `DynamoDbClient`, requires the use of a synchronous HTTP client. The AWS SDK for Java offers three synchronous HTTP clients.

ApacheHttpClient (default)

ApacheHttpClient is the default HTTP client for synchronous service clients. For information about configuring the `ApacheHttpClient`, see Configure the Apache-based HTTP client.

AwsCrtHttpClient

AwsCrtHttpClient provides high throughput and non-blocking IO. It is built on the AWS Common Runtime (CRT) Http Client. For information about configuring the `AwsCrtHttpClient` and using it with service clients, see the section called "Configure AWS CRT-based HTTP clients".

UrlConnectionHttpClient

To minimize the number of jars and third-party libraries you application uses, you can use the UrlConnectionHttpClient. For information about configuring the `UrlConnectionHttpClient`, see Configure the URLConnection-based HTTP client.

Asynchronous clients

Asynchronous HTTP clients in the SDK for Java implement the SdkAsyncHttpClient interface. An asynchronous service client, such as the `S3AsyncClient` or the `DynamoDbAsyncClient`, requires the use of an asynchronous HTTP client. The AWS SDK for Java offers two asynchronous HTTP clients.

NettyNioAsyncHttpClient (default)

NettyNioAsyncHttpClient is the default HTTP client used by asynchronous clients. For information about configuring the `NettyNioAsyncHttpClient`, see the section called “Configure the Netty-based HTTP client”.

AwsCrtAsyncHttpClient

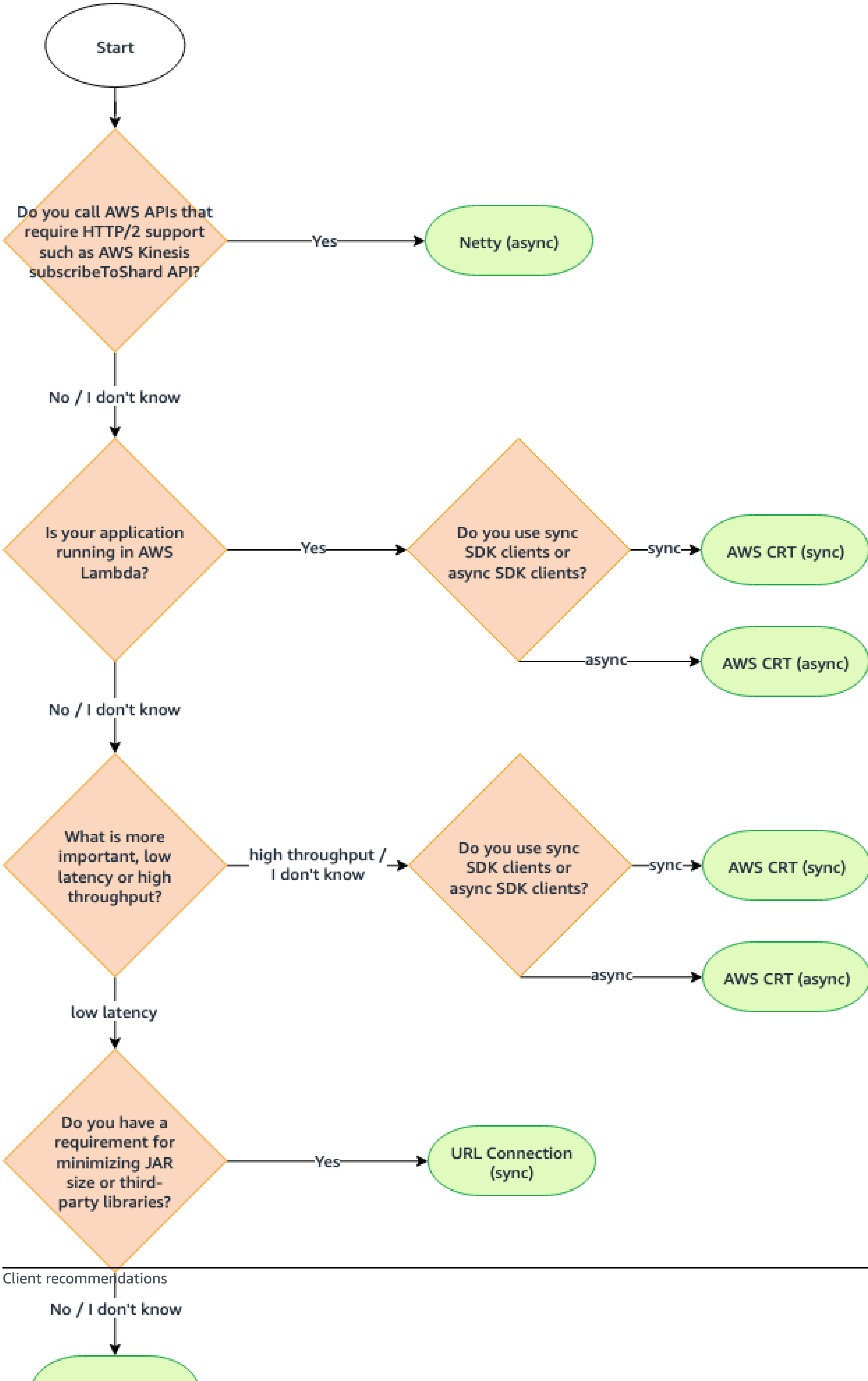
The AwsCrtAsyncHttpClient is based on the AWS Common Runtime (CRT) HTTP Client. For information about configuring the `AwsCrtAsyncHttpClient`, see the section called “Configure AWS CRT-based HTTP clients”.

HTTP client recommendations

Several factors come into play when you choose an HTTP client implementation. Use the following information to help you decide.

Recommendation flowchart

The following flowchart provides general guidance to help you determine which HTTP client to use.


Start
Do you call AWS APIs that require HTTP/2 support such as AWS Kinesis subscribeToShard API?
Yes
Netty (async)
No / I don't know
Is your application running in AWS Lambda?
Yes
Do you use sync SDK clients or async SDK clients?
sync
AWS CRT (sync)
async
AWS CRT (async)
No / I don't know
What is more important, low latency or high throughput?
high throughput / I don't know
Do you use sync SDK clients or async SDK clients?
sync
AWS CRT (sync)
async
AWS CRT (async)
low latency
Do you have a requirement for minimizing JAR size or third-party libraries?
Yes
URL Connection (sync)
No / I don't know

HTTP client comparison

The following table provides detailed information for each HTTP client.

HTTP client	Sync or async	When to use	Limitation/ drawback
Apache-based HTTP client *(default sync HTTP client)*	Sync	Use it if you prefer low latency over high throughput	Slower startup time compared to other HTTP clients
URLConnection-based HTTP client	Sync	Use it if you have a hard requireme nt for limiting third-party dependencies	Does not support the HTTP PATCH method, required by some APIS like Amazon APIGatewa y Update operations
AWS CRT-based sync HTTP client[1]	Sync	• Use it if your application is running in AWS Lambda • Use it if you prefer high throughpu t over low latency • Use it if you prefer sync SDK clients	N/A
Netty-based HTTP client *(default async HTTP client)*	Async	• Use it if your application invokes APIs that require HTTP/2 support such as Kinesis API Subscribe ToShard	Slower startup time compared to other HTTP clients

HTTP client	Sync or async	When to use	Limitation/ drawback
AWS CRT-based async HTTP client[1]	Async	• Use it if your application is running in AWS Lambda • Use it if you prefer high throughput over low latency • Use it if you prefer async SDK clients	• Does not support service clients that require HTTP/2 support such as `KinesisAsyncClient` and `TranscribeStreamingAsyncClient`

[1]Because of their added benefits, we recommend that you use the AWS CRT-based HTTP clients if possible.

Smart configuration defaults

The AWS SDK for Java 2.x (version 2.17.102 or later) offers a smart configuration defaults feature. This feature optimizes two HTTP client properties along with other properties that don't affect the HTTP client.

The smart configuration defaults set sensible values for the `connectTimeoutInMillis` and `tlsNegotiationTimeoutInMillis` properties based on a defaults mode value that you provide. You choose the defaults mode value based on your application's characteristics.

For more information about smart configuration defaults and how to choose the defaults mode value that is best suited for your applications, see the AWS SDKs and Tools Reference Guide.

Following are four ways to set the defaults mode for your application.

Service client

Use the service client builder to configure the defaults mode directly on the service client. The following example sets the defaults mode to `auto` for the `DynamoDbClient`.

```
DynamoDbClient ddbClient = DynamoDbClient.builder()
```

```
                                   .defaultsMode(DefaultsMode.AUTO)
                                   .build();
```

System property

You can use the `aws.defaultsMode` system property to specify the defaults mode. If you set the system property in Java, you need to set the property before initializing any service client.

The following example shows you how to set the defaults mode to `auto` using a system property set in Java.

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

The following example demonstrates how you set the defaults mode to `auto` using a `-D` option of the `java` command.

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

Environment variable

Set a value for environment variable `AWS_DEFAULTS_MODE` to select the defaults mode for your application.

The following information shows the command to run to set the value for the defaults mode to `auto` using an environment variable.

Operating system	Command to set environment variables
Linux, macOS, or Unix	`export AWS_DEFAULTS_MODE=auto`
Windows	`set AWS_DEFAULTS_MODE=auto`

AWS config file

You can add a `defaults_mode` configuration property to the shared AWS `config` file as the following example shows.

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

If you set the defaults mode globally with the system property, environment variable, or AWS config file, you can override the settings when you build an HTTP client.

When you build an HTTP client with the `httpClientBuilder()` method, settings apply only to the instance that you are building. An example of this is shown here. The Netty-based HTTP client in this example overrides any default mode values set globally for `connectTimeoutInMillis` and `tlsNegotiationTimeoutInMillis`.

Configure the Apache-based HTTP client

Synchronous service clients in the AWS SDK for Java 2.x use an Apache-based HTTP client, ApacheHttpClient by default. The SDK's `ApacheHttpClient` is based on the Apache HttpClient.

The SDK also offers the UrlConnectionHttpClient, which loads more quickly, but has fewer features. For information about configuring the `UrlConnectionHttpClient`, see the section called “Configure the URLConnection-based HTTP client”.

To see the full set of configuration options available to you for the `ApacheHttpClient`, see ApacheHttpClient.Builder and ProxyConfiguration.Builder.

Access the `ApacheHttpClient`

In most situations, you use the `ApacheHttpClient` without any explicit configuration. You declare your service clients and the SDK will configure the `ApacheHttpClient` with standard values for you.

If you want to explicitly configure the `ApacheHttpClient` or use it with multiple service clients, you need to make it available for configuration.

No configuration needed

When you declare a dependency on a service client in Maven, the SDK adds a *runtime* dependency on the `apache-client` artifact. This makes the `ApacheHttpClient` class available to your code at runtime, but not at compile time. If you are not configuring the Apache-based HTTP client, you do not need to specify a dependency for it.

In the following XML snippet of a Maven `pom.xml` file, the dependency declared with `<artifactId>s3</artifactId>` automatically brings in the Apache-based HTTP client. You don't need to declare a dependency specifically for it.

```
<dependencyManagement>
```

```
    <dependencies>
         <dependency>
             <groupId>software.amazon.awssdk</groupId>
             <artifactId>bom</artifactId>
             <version>2.17.290</version>
             <type>pom</type>
             <scope>import</scope>
         </dependency>
     </dependencies>
</dependencyManagement>
<dependencies>
    <!-- The s3 dependency automatically adds a runtime dependency on the
 ApacheHttpClient-->
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3</artifactId>
    </dependency>
</dependencies>
```

With these dependencies, you cannot make any explicit HTTP configuration changes, because the `ApacheHttpClient` library is only on the runtime classpath.

Configuration needed

To configure the `ApacheHttpClient`, you need to add a dependency on the `apache-client` library at *compile* time.

Refer to the following example of a Maven `pom.xml` file to configure the `ApacheHttpClient`.

```
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>2.17.290</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
```

```
            <artifactId>s3</artifactId>
        </dependency>
        <!-- By adding the apache-client dependency, ApacheHttpClient will be added to
             the compile classpath so you can configure it. -->
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>apache-client</artifactId>
        </dependency>
    </dependencies>
```

Use and configure the `ApacheHttpClient`

You can configure an instance of `ApacheHttpClient` along with building a service client, or you can configure a single instance to share across multiple service clients.

With either approach, you use the [`ApacheHttpClient.Builder`]() to configure the properties for the Apache-based HTTP client.

Best practice: dedicate an `ApacheHttpClient` instance to a service client

If you need to configure an instance of the `ApacheHttpClient`, we recommend that you build the dedicated `ApacheHttpClient` instance. You can do so by using the `httpClientBuilder` method of the service client's builder. This way, the lifecycle of the HTTP client is managed by the SDK, which helps avoid potential memory leaks if the `ApacheHttpClient` instance is not closed down when it's no longer needed.

The following example creates an `S3Client` and configures the embedded instance of `ApacheHttpClient` with `maxConnections` and `connectionTimeout` values. The HTTP instance is created using the `httpClientBuilder` method of `S3Client.Builder`.

Imports

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.time.Duration;
```

Code

```
S3Client s3Client = S3Client   // Singleton: Use the s3Client for all requests.
    .builder()
    .httpClientBuilder(ApacheHttpClient.builder()
```

```
        .maxConnections(100)
        .connectionTimeout(Duration.ofSeconds(5))
    ).build();

// Perform work with the s3Client.

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

Alternative approach: share an `ApacheHttpClient` instance

To help keep resource and memory usage lower for your application, you can configure an `ApacheHttpClient` and share it across multiple service clients. The HTTP connection pool will be shared, which lowers resource usage.

> **Note**
>
> When an `ApacheHttpClient` instance is shared, you must close it when it is ready to be disposed. The SDK will not close the instance when the service client is closed.

The following example configures an Apache-based HTTP client that is used by two service clients. The configured `ApacheHttpClient` instance is passed to the `httpClient` method of each builder. When the service clients and the HTTP client are no longer needed, the code explicitly closes them. The code closes the HTTP client last.

Imports

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
```

Code

```
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()
        .maxConnections(100).build();

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

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

// Perform work with the s3Client and dynamoDbClient.

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

Proxy configuration example

The following code snippet uses the [proxy configuration builder for the Apache HTTP client](#).

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

The equivalent Java system properties for the proxy configuration are shown in the following command line snippet.

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

The equivalent setup that uses environment variables is:

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

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

```
                        .useSystemPropertyValues(Boolean.FALSE)
                        .build())
                .build();

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

> **Note**
>
> The Apache HTTP client does not currently support HTTPS proxy system properties or the HTTPS_PROXY environment variable.

Configure the URLConnection-based HTTP client

The AWS SDK for Java 2.x offers a lighter-weight `UrlConnectionHttpClient` HTTP client in comparison to the default `ApacheHttpClient`. The `UrlConnectionHttpClient` is based on Java's `URLConnection`.

The `UrlConnectionHttpClient` loads more quickly than the Apache-based HTTP client, but has fewer features. Because it loads more quickly, it is a good solution for Java AWS Lambda functions.

The `UrlConnectionHttpClient` has several configurable options that you can access.

> **Note**
>
> The `UrlConnectionHttpClient` does not support the HTTP PATCH method.
> A handful of AWS API operations require PATCH requests. Those operation names usually start with `Update*`. The following are several examples.
>
> - Several `Update*` operations in the AWS Security Hub API and also the BatchUpdateFindings operation
> - All Amazon API Gateway API `Update*` operations
> - Several `Update*` operations in the Amazon WorkDocs API
>
> If you might use the `UrlConnectionHttpClient`, first refer to the API Reference for the AWS service that you're using. Check to see if the operations you need use the PATCH operation.

Access the `UrlConnectionHttpClient`

To configure and use the `UrlConnectionHttpClient`, you declare a dependency on the `url-connection-client` Maven artifact in your `pom.xml` file.

Unlike the `ApacheHttpClient`, the `UrlConnectionHttpClient` is not automatically added to your project, so use must specifically declare it.

The following example of a `pom.xml` file shows the dependencies required to use and configure the HTTP client.

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.17.290</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<!-- other dependencies such as s3 or dynamodb -->

<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>url-connection-client</artifactId>
    </dependency>
</dependencies>
```

Use and configure the `UrlConnectionHttpClient`

You can configure an instance of `UrlConnectionHttpClient` along with building a service client, or you can configure a single instance to share across multiple service clients.

With either approach, you use the UrlConnectionHttpClient.Builder to configure the properties for the URLConnection-based HTTP client.

Best practice: dedicate an UrlConnectionHttpClient instance to a service client

If you need to configure an instance of the UrlConnectionHttpClient, we recommend that you build the dedicated UrlConnectionHttpClient instance. You can do so by using the httpClientBuilder method of the service client's builder. This way, the lifecycle of the HTTP client is managed by the SDK, which helps avoid potential memory leaks if the UrlConnectionHttpClient instance is not closed down when it's no longer needed.

The following example creates an S3Client and configures the embedded instance of UrlConnectionHttpClient with socketTimeout and proxyConfiguration values. The proxyConfiguration method takes a Java lambda expression of type Consumer<ProxyConfiguration.Builder>.

Imports

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.urlconnection.UrlConnectionHttpClient;
import java.net.URI;
import java.time.Duration;
```

Code

```
// Singleton: Use the s3Client for all requests.
S3Client s3Client =
    S3Client.builder()
            .httpClientBuilder(UrlConnectionHttpClient.builder()
                    .socketTimeout(Duration.ofMinutes(5))
                    .proxyConfiguration(proxy -> proxy.endpoint(URI.create("http://
proxy.mydomain.net:8888"))))
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

// Perform work with the s3Client.

s3Client.close();   // Requests completed: Close the s3client.
```

Alternative approach: share an UrlConnectionHttpClient instance

To help keep resource and memory usage lower for your application, you can configure an UrlConnectionHttpClient and share it across multiple service clients. The HTTP connection pool will be shared, which lowers resource usage.

> **Note**
>
> When an `UrlConnectionHttpClient` instance is shared, you must close it when it is ready to be disposed. The SDK will not close the instance when the service client is closed.

The following example configures an URLConnection-based HTTP client that is used by two service clients. The configured `UrlConnectionHttpClient` instance is passed to the `httpClient` method of each builder. When the service clients and the HTTP client are no longer needed, the code explicitly closes them. The code closes the HTTP client last.

Imports

```
import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.urlconnection.ProxyConfiguration;
import software.amazon.awssdk.http.urlconnection.UrlConnectionHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.net.URI;
import java.time.Duration;
```

Code

```
SdkHttpClient urlHttpClient = UrlConnectionHttpClient.create();

// Singletons: Use the s3Client and dynamoDbClient for all requests.
S3Client s3Client =
    S3Client.builder()
            .httpClient(urlHttpClient)
            .defaultsMode(DefaultsMode.IN_REGION)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

DynamoDbClient dynamoDbClient =
    DynamoDbClient.builder()
                  .httpClient(urlHttpClient)
                  .defaultsMode(DefaultsMode.IN_REGION)
                  .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                  .build();
```

```
// Perform work with the s3Client and dynamoDbClient.

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

Use URLConnectionHttpClient and ApacheHttpClient together

When you use the `UrlConnectionHttpClient` in your application, you must supply each service client with either a `URLConnectionHttpClient` instance or a `ApacheHttpClient` instance using the service client builder's `httpClientBuilder` method.

An exception occurs if your program uses multiple service clients and both of the following are true:

- One service client is configured to use a `UrlConnectionHttpClient` instance
- Another service client uses the default `ApacheHttpClient` without explicitly building it with the `httpClient()` or `httpClientBuilder()` methods

The exception will state that multiple HTTP implementations were found on the classpath.

The following example code snippet leads to an exception.

```
// The dynamoDbClient uses the UrlConnectionHttpClient
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
        .httpClient(UrlConnectionHttpClient.create())
        .build();

// The s3Client below uses the ApacheHttpClient at runtime, without specifying it.
// An SdkClientException is thrown with the message that multiple HTTP implementations
 were found on the classpath.
S3Client s3Client = S3Client.create();

// Perform work with the s3Client and dynamoDbClient.

dynamoDbClient.close();
s3Client.close();
```

Avoid the exception by explicitly configuring the `S3Client` with an `ApacheHttpClient`.

```
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
        .httpClient(UrlConnectionHttpClient.create())
        .build();

S3Client s3Client = S3Client.builder()
        .httpClient(ApacheHttpClient.create())    // Explicitly build the
 ApacheHttpClient.
        .build();

// Perform work with the s3Client and dynamoDbClient.

dynamoDbClient.close();
s3Client.close();
```

Note

To explicitly create the `ApacheHttpClient`, you must add a dependency on the `apache-client` artifact in your Maven project file.

Proxy configuration example

The following code snippet uses the proxy configuration builder for the URL connection HTTP client.

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

The equivalent Java system properties for the proxy configuration are shown in the following command line snippet.

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

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

The equivalent setup that uses environment variables is:

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

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

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

Note

The URLConnection-based HTTP client does not currently support HTTPS proxy system properties or the HTTPS_PROXY environment variable.

Configure the Netty-based HTTP client

The default HTTP client for asynchronous operations in the AWS SDK for Java 2.x is the Netty-based NettyNioAsyncHttpClient. The Netty-based client is based on the asynchronous event-driven network framework of the Netty project.

As an alternative HTTP client, you can use the new AWS CRT-based HTTP client. This topic shows you how to configure the `NettyNioAsyncHttpClient`.

Access the `NettyNioAsyncHttpClient`

In most situations, you use the `NettyNioAsyncHttpClient` without any explicit configuration in asynchronous programs. You declare your asynchronous service clients and the SDK will configure the `NettyNioAsyncHttpClient` with standard values for you.

If you want to explicitly configure the `NettyNioAsyncHttpClient` or use it with multiple service clients, you need to make it available for configuration.

No configuration needed

When you declare a dependency on a service client in Maven, the SDK adds a *runtime* dependency on the `netty-nio-client` artifact. This makes the `NettyNioAsyncHttpClient` class available to your code at runtime, but not at compile time. If you are not configuring the Netty-based HTTP client, you don't need to specify a dependency for it.

In the following XML snippet of a Maven `pom.xml` file, the dependency declared with `<artifactId>dynamodb-enhanced</artifactId>` transitively brings in the Netty-based HTTP client. You don't need to declare a dependency specifically for it.

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.17.290</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>dynamodb-enhanced</artifactId>
    </dependency>
</dependencies>
```

With these dependencies, you cannot make any HTTP configuration changes, since the `NettyNioAsyncHttpClient` library is only on the runtime classpath.

Configuration needed

To configure the `NettyNioAsyncHttpClient`, you need to add a dependency on the `netty-nio-client` artifact at *compile* time.

Refer to the following example of a Maven `pom.xml` file to configure the `NettyNioAsyncHttpClient`.

```
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>2.17.290</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>dynamodb-enhanced</artifactId>
        </dependency>
        <!-- By adding the netty-nio-client dependency, NettyNioAsyncHttpClient will
 be
             added to the compile classpath so you can configure it. -->
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>netty-nio-client</artifactId>
        </dependency>
    </dependencies>
```

Use and configure the `NettyNioAsyncHttpClient`

You can configure an instance of `NettyNioAsyncHttpClient` along with building a service client, or you can configure a single instance to share across multiple service clients.

With either approach, you use the NettyNioAsyncHttpClient.Builder to configure the properties for the Netty-based HTTP client instance.

Best practice: dedicate a `NettyNioAsyncHttpClient` instance to a service client

If you need to configure an instance of the `NettyNioAsyncHttpClient`, we recommend that you build a dedicated `NettyNioAsyncHttpClient` instance. You can do so by using the `httpClientBuilder` method of the service client's builder. This way, the lifecycle of the HTTP client is managed by the SDK, which helps avoid potential memory leaks if the `NettyNioAsyncHttpClient` instance is not closed down when it's no longer needed.

The following example creates a `DynamoDbAsyncClient` instance that is used by a `DynamoDbEnhancedAsyncClient` instance. The `DynamoDbAsyncClient` instance contains the `NettyNioAsyncHttpClient` instance with `connectionTimeout` and `maxConcurrency` values. The HTTP instance is created using `httpClientBuilder` method of `DynamoDbAsyncClient.Builder`.

Imports

```
import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedAsyncClient;
import
 software.amazon.awssdk.enhanced.dynamodb.extensions.AutoGeneratedTimestampRecordExtension;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.time.Duration;
```

Code

```
// DynamoDbAsyncClient is the lower-level client used by the enhanced client.
DynamoDbAsyncClient dynamoDbAsyncClient =
    DynamoDbAsyncClient
        .builder()
            .httpClientBuilder(NettyNioAsyncHttpClient.builder()
            .connectionTimeout(Duration.ofMillis(5_000))
            .maxConcurrency(100)
            .tlsNegotiationTimeout(Duration.ofMillis(3_500)))
        .defaultsMode(DefaultsMode.IN_REGION)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();

// Singleton: Use dynamoDbAsyncClient and enhancedClient for all requests.
DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient
        .builder()
        .dynamoDbClient(dynamoDbAsyncClient)
        .extensions(AutoGeneratedTimestampRecordExtension.create())
        .build();

// Perform work with the dynamoDbAsyncClient and enhancedClient.

// Requests completed: Close dynamoDbAsyncClient.
```

```
dynamoDbAsyncClient.close();
```

Alternative approach: share a `NettyNioAsyncHttpClient` instance

To help keep resource and memory usage lower for your application, you can configure a `NettyNioAsyncHttpClient` and share it across multiple service clients. The HTTP connection pool will be shared, which lowers resource usage.

> **Note**
>
> When a `NettyNioAsyncHttpClient` instance is shared, you must close it when it is ready to be disposed. The SDK will not close the instance when the service client is closed.

The following example configures a Netty-based HTTP client that is used by two service clients. The configured `NettyNioAsyncHttpClient` instance is passed to the `httpClient` method of each builder. When the service clients and the HTTP client are no longer needed, the code explicitly closes them. The code closes the HTTP client last.

Imports

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
```

Code

```
// Create a NettyNioAsyncHttpClient shared instance.
SdkAsyncHttpClient nettyHttpClient =
 NettyNioAsyncHttpClient.builder().maxConcurrency(100).build();

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

DynamoDbAsyncClient dbAsyncClient =
```

```
    DynamoDbAsyncClient.builder()
                        .httpClient(nettyHttpClient)
                        .defaultsMode(DefaultsMode.IN_REGION)

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

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

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

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

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

Proxy configuration example

The following code snippet uses the proxy configuration builder for the Netty HTTP client.

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

The equivalent Java system properties for the proxy configuration are shown in the following command line snippet.

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

Important

To use any of the HTTPS proxy system properties, the `scheme` property must be set in code to `https`. If the scheme property is not set in code, the scheme defaults to HTTP and the SDK looks only for `http.*` system properties.

The equivalent setup that uses environment variables is:

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

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

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

Configure AWS CRT-based HTTP clients

The AWS CRT-based HTTP clients include the synchronous [AwsCrtHttpClient] and asynchronous [AwsCrtAsyncHttpClient]. The AWS CRT-based HTTP clients provide the following HTTP client benefits:

- Faster SDK startup time
- Smaller memory footprint
- Reduced latency time
- Connection health management
- DNS load balancing

AWS CRT-based components in the SDK

The AWS CRT-based *HTTP* clients, described in this topic, and the AWS CRT-based *S3* client are different components in the SDK.

The synchronous and asynchronous **AWS CRT-based HTTP clients** are implementations SDK HTTP client interfaces and are used for general HTTP communication. They are alternatives to the other synchronous or asynchronous HTTP clients in the SDK with additional benefits.

The **AWS CRT-based S3 client** is an implementation of the S3AsyncClient interface and is used for working with the Amazon S3 service. It is an alternative to the Java-based implementation of the `S3AsyncClient` interface and offers several advantages.

Although both components use libraries from the AWS Common Runtime, the AWS CRT-based HTTP clients do not use the aws-c-s3 library and do not support the S3 multipart upload API features. The AWS CRT-based S3 client, by contrast, was purpose-built to support the S3 multipart upload API features.

Access the AWS CRT-based HTTP clients

Before you can use the AWS CRT-based HTTP clients, add the `aws-crt-client` artifact with a minimum version of 2.22.0 to your project's dependencies.

The following Maven `pom.xml` shows the AWS CRT-based HTTP clients declared using the bill of materials (BOM) mechanism.

```
<project>
   <properties>
     <aws.sdk.version>2.22.0</aws.sdk.version>
  </properties>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
   </dependency>
  </dependencies>
```

```
</project>
```

Visit the Maven central repository for the latest version.

Use and configure an AWS CRT-based HTTP client

You can configure an AWS CRT-based HTTP client along with building a service client, or you can configure a single instance to share across multiple service clients.

With either approach, you use a builder to configure the properties for the AWS CRT-based HTTP client instance.

Best practice: dedicate an instance to a service client

If you need to configure an instance of an AWS CRT-based HTTP client, we recommend that you dedicate the instance by building it along with the service client . You can do so by using the `httpClientBuilder` method of the service client's builder. This way, the lifecycle of the HTTP client is managed by the SDK, which helps avoid potential memory leaks if the AWS CRT-based HTTP client instance is not closed down when it's no longer needed.

The following example creates an S3 service client and configures an AWS CRT-based HTTP client with `connectionTimeout` and `maxConcurrency` values.

Synchronous client

Imports

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

Code

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

```
// Perform work with the s3Client.

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

Asynchronous client

Imports

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

Code

```
// Singleton: Use s3AsyncClient for all requests.
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
    .httpClientBuilder(AwsCrtAsyncHttpClient
        .builder()
        .connectionTimeout(Duration.ofSeconds(3))
        .maxConcurrency(100))
    .build();

// Perform work with the s3AsyncClient.

// Requests completed: Close the s3AsyncClient.
s3AsyncClient.close();
```

Alternative approach: share an instance

To help keep resource and memory usage lower for your application, you can configure an AWS CRT-based HTTP client and share it across multiple service clients. The HTTP connection pool will be shared, which lowers resource usage.

> **Note**
>
> When an AWS CRT-based HTTP client instance is shared, you must close it when it is ready to be disposed. The SDK will not close the instance when the service client is closed.

The following example configures an AWS CRT-based HTTP client instance with `connectionTimeout` and `maxConcurrency` values. The configured instance is passed to the `httpClient` method of each service client's builder. When the service clients and the HTTP client are no longer needed, they are explicitly closed. The HTTP client is closed last.

Synchronous client

Imports

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.time.Duration;
```

Code

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

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

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

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

Asynchronous client

Imports

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import java.time.Duration;
```

Code

```
// Create an AwsCrtAsyncHttpClient shared instance.
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(3))
    .maxConcurrency(100)
    .build();

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

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

// Requests completed: Close all service clients.
```

```
s3AsyncClient.close();
dynamoDbAsyncClient.close();
crtAsyncHttpClient.close();  // Explicitly close crtAsyncHttpClient.
```

Set an AWS CRT-based HTTP client as the default

You can setup your Maven build file to have the SDK use an AWS CRT-based HTTP client as the default HTTP client for service clients.

You do this by adding an `exclusions` element with the default HTTP client dependencies to each service client artifact.

In the following `pom.xml` example, the SDK uses an AWS CRT-based HTTP client for S3 services. If the service client in your code is an `S3AsyncClient`, the SDK uses `AwsCrtAsyncHttpClient`. If the service client is an S3Client, the SDK uses `AwsCrtHttpClient`. With this setup the default Netty-based asynchronous HTTP client and the default Apache-based synchronous HTTP are not available.

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

```
    </dependency>
  </dependencies>
</project>
```

Visit the Maven central repository for the latest *VERSION* value.

> **Note**
>
> If multiple service clients are declared in a `pom.xml` file, all require the `exclusions` XML element.

Use a Java system property

To use the AWS CRT-based HTTP clients as the default HTTP for your application, you can set the Java system property `software.amazon.awssdk.http.async.service.impl` to a value of `software.amazon.awssdk.http.crt.AwsCrtSdkHttpService`.

To set during application startup, run a command similar to the following.

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

Use the following code snippet to set the system property in your application code.

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

> **Note**
>
> You need to add a dependency on the `aws-crt-client` artifact in your `poml.xml` file when you use a system property to configure the use of the AWS CRT-based HTTP clients.

Advanced configuration of AWS CRT-based HTTP clients

You can use various configuration settings of the AWS CRT-based HTTP clients, including connection health configuration and maximum idle time. You can review the configuration options available for the AwsCrtAsyncHttpClient. You can configure the same options for the AwsCrtHttpClient.

Connection health configuration

You can configure connection health configuration for the AWS CRT-based HTTP clients by using the connectionHealthConfiguration method on the HTTP client builder.

The following example creates an S3 service client that uses a AWS CRT-based HTTP client instance configured with connection health configuration and a maximum idle time for connections.

Synchronous client

Imports

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

Code

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

// Perform work with s3Client.

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

Asynchronous client

Imports

```
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;
```

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

Code

```
// Singleton: Use the s3AsyncClient for all requests.
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
    .httpClientBuilder(AwsCrtAsyncHttpClient
        .builder()
        .connectionHealthConfiguration(builder -> builder
            .minimumThroughputInBps(32000L)
            .minimumThroughputTimeout(Duration.ofSeconds(3)))
        .connectionMaxIdleTime(Duration.ofSeconds(5)))
    .build();

// Perform work with s3AsyncClient.

// Requests complete: Close the service client.
s3AsyncClient.close();
```

HTTP/2 support

The HTTP/2 protocol is not yet supported in the AWS CRT-based HTTP clients, but is planned for a future release.

In the meantime, if you are using service clients that require HTTP/2 support such as the KinesisAsyncClient or the TranscribeStreamingAsyncClient, consider using the NettyNioAsyncHttpClient instead.

Proxy configuration example

The following code snippet shows the use of the `ProxyConfiguration.Builder` that you use to configure proxy setting in code.

Synchronous client

Imports

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

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

Code

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

Asynchronous client

Imports

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

Code

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

The equivalent Java system properties for the proxy configuration are shown in the following command line snippet.

```
$ java -Dhttps.proxyHost=myproxy -Dhttps.proxyPort=1234 -Dhttps.proxyUser=username \
```

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

> **Important**
>
> To use any of the HTTPS proxy system properties, the `scheme` property must be set in code to `https`. If the scheme property is not set in code, the scheme defaults to HTTP and the SDK looks only for `http.*` system properties.

The equivalent setup that uses environment variables is:

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

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

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

Configure HTTP proxies

You can configure HTTP proxies by using code, by setting Java system properties, or by setting environment variables.

Configure in code

You configure proxies in code with a client-specific `ProxyConfiguration` builder when you build the service client. The following code shows an example proxy configuration for an Apache-based HTTP client that is used by an Amazon S3 service client.

```
SdkHttpClient httpClient1 = ApacheHttpClient.builder()
```

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

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

The section for each HTTP client in this topic shows a proxy configuration example.

- Apache HTTP client
- URLConnection-based HTTP client
- Netty-based HTTP client
- AWS CRT-based HTTP client

Configure HTTP proxies with external settings

Even if you don't explicitly use a `ProxyConfiguration` builder in code, the SDK looks for external settings to configure a default proxy configuration.

By default, the SDK first searches for JVM system properties. If even one property is found, the SDK uses the value and any other system property values. If no system properties are available, the SDK looks for proxy environment variables.

The SDK can use the following Java system properties and environment variables.

Java system properties

System property	Description	HTTP client support
http.proxyHost	Host name of the HTTP proxy server	All
http.proxyPort	Port number of the HTTP proxy server	All

System property	Description	HTTP client support
http.proxyUser	Username for HTTP proxy authentication	All
http.proxyPassword	Password for HTTP proxy authentication	All
http.nonProxyHosts	List of hosts that should be reached directly, bypassing the proxy. This list is also valid when HTTPS is used.	All
https.proxyHost	Host name of the HTTPS proxy server	Netty, CRT
https.proxyPort	Port number of the HTTPS proxy server	Netty, CRT
https.proxyUser	Username for HTTPS proxy authentication	Netty, CRT
https.proxyPassword	Password for HTTPS proxy authentication	Netty, CRT

Environment variables

Environment variable	Description	HTTP client support
HTTP_PROXY[1]	A valid URL with a scheme of HTTP	All
HTTPS_PROXY[1]	A valid URL with a scheme of HTTPS	Netty, CRT
NO_PROXY[2]	List of hosts that should be reached directly, bypassing the proxy. The list is valid for both HTTP and HTTPS.	All

View key and footnotes

All - All HTTP clients offered by the SDK–`UrlConnectionHttpClient`, `ApacheHttpClient`, `NettyNioAsyncHttpClient`, `AwsCrtAsyncHttpClient`.

Netty - The Netty-based HTTP client (`NettyNioAsyncHttpClient`).

CRT - The AWS CRT-based HTTP clients, (`AwsCrtHttpClient` and `AwsCrtAsyncHttpClient`).

[1]The environment variable queried, whether `HTTP_PROXY` or `HTTPS_PROXY`, depends on the scheme setting in the client's `ProxyConfiguration`. The default scheme is HTTP. The following snippet shows how to change the scheme to HTTPS used for environment variable resolution.

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

[2]The `NO_PROXY` environment variable supports a mix of "|" and "," separators between host names. Host names may include the "*" wildcard.

Use a combination of settings

You can use a combination of HTTP proxy settings in code, system properties, and environment variables.

Example – configuration provided by a system property and by code

```
// Command line with the proxy password set as a system property.
$ java -Dhttp.proxyPassword=SYS_PROP_password -cp ... App

// Since the 'useSystemPropertyValues' setting is 'true' (the default), the SDK will
 supplement
// the proxy configuration in code with the 'http.proxyPassword' value from the system
 property.
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()
            .proxyConfiguration(ProxyConfiguration.builder()
                    .endpoint(URI.create("http://localhost:1234"))
                    .username("username")
                    .build())
```

```
            .build();

// Use the apache HTTP client with proxy configuration.
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
                    .httpClient(apacheHttpClient)
                    .build();
```

The SDK resolves the following proxy settings.

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

Example – both system properties and environment variables are available

Each HTTP client's `ProxyConfiguration` builder offers settings named `useSystemPropertyValues` and `useEnvironmentVariablesValues`. By default, both setting are `true`. When `true`, the SDK automatically uses values from system properties or environment variables for options that are not provided by the `ProxyConfiguration` builder.

> **Important**
>
> System properties take precedence over environment variables. If an HTTP proxy system property is found, the SDK retrieves ***all*** values from system properties and none from environment variables. If you want to prioritize environment variables over system properties, set `useSystemPropertyValues` to `false`.

For this example, the following settings are available a runtime:

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

// Environment variables
HTTP_PROXY="http://EnvironmentUser:EnvironmentPassword@ENV_VAR_HOST:3333"
```

```
NO_PROXY="environmentnonproxy.host,environmentnonproxy2.host:1234"
```

The service client is created with one of the following statements. None of the statements explicitly set a proxy setting.

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

The following proxy settings are resolved by the SDK:

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

Because the service client has default proxy settings, the SDK searches for system properties and then environment variables. Since system properties settings take precedence over environment variables, the SDK uses only system properties.

If the use of system properties is changed to `false` as shown in the following code, the SDK resolves only the environment variables.

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

The resolved proxy settings using HTTP are:

```
Host = ENV_VAR_HOST
```

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

Exception handling for the AWS SDK for Java 2.x

Understanding how and when the AWS SDK for Java 2.x throws exceptions is important to building high-quality applications using the SDK. The following sections describe the different cases of exceptions that are thrown by the SDK and how to handle them appropriately.

Why unchecked exceptions?

The AWS SDK for Java uses runtime (or unchecked) exceptions instead of checked exceptions for these reasons:

- To allow developers fine-grained control over the errors they want to handle without forcing them to handle exceptional cases they aren't concerned about (and making their code overly verbose)
- To prevent scalability issues inherent with checked exceptions in large applications

In general, checked exceptions work well on small scales, but can become troublesome as applications grow and become more complex.

AwsServiceException (and subclasses)

AwsServiceException is the most common exception that you'll experience when using the AWS SDK for Java. `AwsServiceException` is a subclass of the more general SdkServiceException. `AwsServiceExceptions` represent an error response from an AWS service. For example, if you try to terminate an Amazon EC2 instance that doesn't exist, Amazon EC2 will return an error response and all the details of that error response will be included in the `AwsServiceException` that's thrown.

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

`AwsServiceException` provides you with information such as:

- Returned HTTP status code
- Returned AWS error code
- Detailed error message from the service in the AwsErrorDetails class
- AWS request ID for the failed request

In most cases, a service-specific subclass of `AwsServiceException` is thrown to allow developers fine-grained control over handling error cases through catch blocks. The Java SDK API reference for AwsServiceException displays the large number of `AwsServiceException` subclasses. Use the subclass links to drill down to see the granular exceptions thrown by a service.

For example, the following links to the SDK API reference show the exception hierarchies for a few common AWS services. The list of subclasses shown on each pages shows the specific exceptions that your code can catch.

- Amazon S3
- DynamoDB
- Amazon SQS

To learn more about an exception, inspect the `errorCode` on the AwsErrorDetails object. You can use the `errorCode` value to look up information in the service guide API. For example if an `S3Exception` is caught and the `AwsErrorDetails#errorCode()` value is `InvalidRequest`, use the list of error codes in the Amazon S3 API Reference to see more details.

SdkClientException

SdkClientException indicates that a problem occurred inside the Java client code, either while trying to send a request to AWS or while trying to parse a response from AWS. An `SdkClientException` is generally more severe than an `SdkServiceException`, and indicates a major problem that is preventing the client from making service calls to AWS services. For example, the AWS SDK for Java throws an `SdkClientException` if no network connection is available when you try to call an operation on one of the clients.

Exceptions and retry behavior

The SDK for Java retries requests for several client-side exceptions and for HTTP status codes that it receives from AWS service responses. These errors are handled as part of the legacy `RetryMode` that service clients use by default. The Java API reference for `RetryMode` describes the various ways that you can configure the mode.

To customize the exceptions and HTTP status codes that trigger automatic retries, configure your service client with a `RetryPolicy` that adds `RetryOnExceptionsCondition` and `RetryOnStatusCodeCondition` instances.

Retries

Calls to AWS services can fail occasionally for unexpected reasons. Certain errors, such as throttling (rate exceeded) or transient errors, might succeed if the call is retried. The AWS SDK for Java 2.x has a built-in mechanism to detect such errors and automatically retry the call that is enabled by default for all clients.

This page describes how this works, how to configure the distinct modes, and tailor the retry behavior.

Retry strategies

A retry strategy is a mechanism used in the SDK to implement retries. Each SDK client has a retry strategy created at build time that cannot be modified after the client is built.

The retry strategy has the following responsibilities.

- Classify exceptions as retryable or not.
- Compute the suggested delay to wait before the next attempt.
- Maintain a token bucket that provides a mechanism to stop retries when a large percentage of requests are failing and retries are unsuccessful.

> **Note**
>
> Before the release of retry *strategies* with version 2.26.0 of the SDK, retry *policies* provided the retry mechanism in the SDK. The retry *policy* API is made up of the core `RetryPolicy` class in the `software.amazon.awssdk.core.retry` package, whereas the `software.amazon.awssdk.retries` package contains the retry *strategy* API elements.
> The retry strategy API was introduced as part of the AWS-wide effort to unify the interfaces and behavior of the core components of the SDKs.

The SDK for Java 2.x has three built-in retry strategies: standard, legacy, and adaptive. All three retry strategies are preconfigured to retry on a set of retryable exceptions. Examples of retryable errors are socket timeouts, service-side throttling, concurrency or optimistic lock failures, and transient service errors.

Standard retry strategy

The standard retry strategy is the recommended `RetryStrategy` implementation for normal use cases. Unlike the `AdaptiveRetryStrategy`, the standard strategy is generally useful across all retry use cases.

By default, the standard retry strategy does the following.

- Retries on the conditions that are configured at build time. You can adjust this with `StandardRetryStrategy.Builder#retryOnException`.
- Retries 2 times for a total of 3 attempts. You can adjust this with `StandardRetryStrategy.Builder#maxAttempts(int)`.
- Uses the `BackoffStrategy#exponentialDelay` backoff strategy, with a base delay of 100 milliseconds and a max delay of 20 seconds. You can adjust with `StandardRetryStrategy.Builder#backoffStrategy`.
- Performs circuit breaking (disabling retries) in the event of high downstream failures. The first attempt is always executed, only retries are disabled. Adjust with `StandardRetryStrategy.Builder#circuitBreakerEnabled`.

Legacy retry strategy

The legacy retry strategy is a `RetryStrategy` for normal use cases, however, it is deprecated in favor of the `StandardRetryStrategy`. This is the default retry strategy used by clients when you don't specify another strategy.

It is characterized by treating throttling and non-throttling exceptions differently, for throttling exceptions the base delay for the backoff is larger (500ms) than the base delay for non-throttling exceptions (100ms), and throttling exceptions do not affect the token bucket state.

Experience using this strategy at scale inside AWS has shown that is not particularly better than the standard retry strategy. Moreover, it fails to protect downstream services from retry storms and can lead to resource starvation on the client side.

By default, the legacy retry strategy does the following.

- Retries on the conditions that are configured at build time. You can adjust this with `LegacyRetryStrategy.Builder#retryOnException` .
- Retries 3 times for a total of 4 attempts. You can adjust this with `LegacyRetryStrategy.Builder#maxAttempts(int)`.
- For non-throttling exceptions, it uses the `BackoffStrategy#exponentialDelay` backoff strategy, with a base delay of 100 milliseconds and a max delay of 20 seconds. You can adjust this with `RetryStrategy.Builder#backoffStrategy.`
- For throttling exceptions, it uses the `BackoffStrategy#exponentialDelay` backoff strategy, with a base delay of 500 milliseconds and a max delay of 20 seconds. You can adjust this with `LegacyRetryStrategy.Builder#throttlingBackoffStrategy`.
- Performs circuit breaking (disabling retries) in the event of high downstream failures. Circuit breaking never prevents a successful first attempt. You can adjust this behavior with `LegacyRetryStrategy.Builder#circuitBreakerEnabled`.
- The state of the circuit breaker is not affected by throttling exceptions.

Adaptive retry strategy

The adaptive retry strategy is a `RetryStrategy` for use cases with a high level of resource constraints.

The adaptive retry strategy includes all the features of the standard strategy and adds a client-side rate limiter that measures the rate of throttled requests compared to non-throttled requests. The strategy uses this measurement to slow down the requests in an attempt to stay within a safe bandwidth, ideally causing zero throttling errors.

By default, the adaptive retry strategy does the following.

- Retries on the conditions that are configured at build time. You can adjust this with `AdaptiveRetryStrategy.Builder#retryOnException` .
- Retries 2 times for a total of 3 attempts. You can adjust this with `AdaptiveRetryStrategy.Builder#maxAttempts(int)`.
- Uses a dynamic backoff delay that is based on the current load against the downstream resource.
- Performs circuit breaking (disabling retries) when there are high number of downstream failures. Circuit breaking may prevent a second attempt in outage scenarios to protect the downstream service.

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

> **Important**
>
> The release of retry strategies with 2.26.0 of the Java SDK includes the new `RetryMode.ADAPTIVE_V2` enumeration value. The `ADAPTIVE_V2` mode corrects an error that failed to delay the first attempt when throttling errors were detected previously.
> With the 2.26.0 release, users automatically get the `ADAPTIVE_V2` mode behavior by setting the mode as `adaptive` with an environment variable, system property, or profile setting. There is no `adaptive_v2` value for these settings. See the following the section called "Specify a strategy" section for how to set the mode.
> Users can get the previous behavior by setting the mode in code using `RetryMode.ADAPTIVE`.

Summary: Comparison of retry strategy default values

The following table shows the default values for the properties of each retry strategy.

Strategy	Maximum attempts	Base delay for non-throttling errors	Base delay for throttling errors	Token bucket size	Token cost per non-throttling retry	Token cost per throttling retry
Standard	3	100	100	500	5	5
Legacy	4	100	500	500	5	0

Strategy	Maximum attempts	Base delay for non-throttling errors	Base delay for throttling errors	Token bucket size	Token cost per non-throttling retry	Token cost per throttling retry
Adaptive	3	100	100	500	5	5

Specify a strategy

You have four ways to specify a strategy for your service client.

In code

When you build a client, you can configure a lambda expression with a retry strategy. The following snippet configures a standard retry strategy that uses default values on a DynamoDB service client.

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

You can specify `RetryMode.LEGACY` or `RetryMode.ADAPTIVE` in place of `RetryMode.STANDARD`.

As a profile setting

Include `retry_mode` as profile setting in the shared AWS config file. Specify `standard`, `legacy`, or `adaptive` as a value. When set as a profile setting, all service clients that are created while the profile is active use the specified retry strategy with default values. You can override this setting by configuring a retry strategy in code as shown previously.

With the following profile, all service clients use the standard retry strategy.

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

As a JVM system property

You can configure a retry stategy for all service clients, unless overridden in code, by using the system property `aws.retryMode`. Specify `standard`, `legacy`, or `adaptive` as a value.

Use the `-D` switch when you invoke Java as shown in the following command.

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

Alternatively, set the system property in the code *before* creating any client as shown in the following snippet.

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

With an environment variable

You can also use the `AWS_RETRY_MODE` environment variable with a value of `standard`, `legacy`, or `adaptive`. As with a profile setting or JVM system property, the environment variable configures all service clients with the specified retry mode unless you configure a client in code.

The following command sets the retry mode to `standard` for the current shell session.

```
export AWS_RETRY_MODE=standard
```

Customize a strategy

You can customize any retry strategy by setting the maximum attempts, backoff strategy, and exceptions that are retryable. You can customize when you build a retry strategy or when you build a client by using a override builder that allows further refinements of the configured strategy.

Customize maximum attempts

You can configure the maximum number of attempts during client construction as shown in the following statement. The following statement customizes the default retry strategy for the client to a maximum of 5 attempt--a first attempt plus 4 retries.

```
DynamoDbClient client = DynamoDbClient.builder()
```

```
        .overrideConfiguration(o -> o.retryStrategy(b -> b.maxAttempts(5)))
        .build();
```

Alternatively, you can build the strategy and provide it to the client as in the following code example. The following code replaces the standard 3 maximum attempts with 10 and configures a DynamoDB client with the customized strategy.

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

> **Warning**
>
> We recommended that you configure each client with a unique `RetryStrategy` instance. If a `RetryStrategy` instance is shared, failures in one client might affect the retry behavior in the other.

You can also set the maximum number of attempts for all clients by using external settings instead of code. You configure this setting as describe in the the section called "Specify a strategy" section.

Customize retryable exceptions

You can configure additional exceptions that trigger retires during client construction. This customization is provided for edge cases where exceptions are thrown that are not included in the default set of retryable exceptions.

The following code snippet shows the methods you use to customize the retry exceptions--`retryOnException` and `retryOnExceptionOrCause`. The `retryOnExceptionOrCause` methods adds a retryable exception if the SDK throws the direct exception or if the exception is wrapped.

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

```
        .build();
```

Customize the backoff strategy

You can build the backoff strategy and supply it to the client.

The following code builds a `BackoffStrategy` that replaces the default, standard strategy's exponential delay backoff strategy.

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

Migrating from `RetryPolicy` to `RetryStrategy`

`RetryPolicy` (the retry policy API) will be supported for the foreseeable future. If you currently use an instance of `RetryPolicy` to configure your client, everything is going to work as before. Behind the scenes the Java SDK adapts it to a `RetryStrategy`. The new retry strategy interfaces provide the same functionality as a `RetryPolicy` but are created and configured differently.

Logging with the SDK for Java 2.x

The AWS SDK for Java 2.x uses 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 2, among others. This topic shows you how to use Log4j 2 as the logging system for working with the SDK.

Log4j 2 configuration file

You typically use a configuration file, named`log4j2.xml` with Log4j 2. Example configuration files are shown below. To learn more about the values used in the configuration file, see the manual for Log4j configuration.

The `log4j2.xml` file needs to be on the classpath when your application starts up. For a Maven project, put the file in the `<project-dir>/src/main/resources` directory.

The `log4j2.xml` configuration file specifies properties such as logging level, where logging output is sent (for example, to a file or to the console), and the format of the output. The logging level specifies the level of detail that Log4j 2 outputs. Log4j 2 supports the concept of multiple logging *hierarchies*. The logging level is set independently for each hierarchy. The main logging hierarchy that you use with the AWS SDK for Java 2.x is `software.amazon.awssdk`.

Add logging dependency

To configure the Log4j 2 binding for SLF4J in your build file, use the following.

Maven

Add the following elements to your `pom.xml` file.

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

Gradle–Kotlin DSL

Add the following to your `build.gradle.kts` file.

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

Use `2.20.0` for the minimum version of the `log4j-slf4j2-impl` artifact. For the latest version, use the version published to Maven central. Replace *`VERSION`* with version you'll use.

SDK-specific errors and warnings

We recommend that you always leave the "software.amazon.awssdk" logger hierarchy set to "WARN" to catch any important messages from the SDK's client libraries. For example, if the

Amazon S3 client detects that your application hasn't properly closed an InputStream and could be leaking resources, the S3 client reports it through a warning message to the logs. This also ensures that messages are logged if the client has any problems handling requests or responses.

The following log4j2.xml file sets the rootLogger to "WARN", which causes warning and error-level messages from all loggers in the application to be output, *including* those in the "software.amazon.awssdk" hierarchy. Alternatively, you can explicitly set the "software.amazon.awssdk" logger hierarchy to "WARN" if <Root level="ERROR"> is used.

Example Log4j2.xml configuration file

This configuration will log messages at the "ERROR" and "WARN" levels to the console for all logger hierarchies.

```
<Configuration status="WARN">
 <Appenders>
  <Console name="ConsoleAppender" target="SYSTEM_OUT">
   <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
  </Console>
 </Appenders>

 <Loggers>
  <Root level="WARN">
   <AppenderRef ref="ConsoleAppender"/>
  </Root>
 </Loggers>
</Configuration>
```

Request/response summary logging

Every request to an AWS service generates a unique AWS request ID that is useful if you run into an issue with how an AWS service is handling a request. AWS request IDs are accessible programmatically through SdkServiceException objects in the SDK for any failed service call, and can also be reported through the "DEBUG" log level of the "software.amazon.awssdk.request" logger.

The following log4j2.xml file enables a summary of requests and responses.

```
<Configuration status="WARN">
 <Appenders>
  <Console name="ConsoleAppender" target="SYSTEM_OUT">
```

```
    <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
   </Console>
  </Appenders>

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

Here is an example of the log output:

```
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Sending Request:
 DefaultSdkHttpFullRequest(httpMethod=POST, protocol=https, host=dynamodb.us-
east-1.amazonaws.com, encodedPath=/, headers=[amz-sdk-invocation-id, Content-Length,
 Content-Type, User-Agent, X-Amz-Target], queryParameters=[])
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Received
 successful response: 200, Request ID:
 QS9DUMME2NHEDH8TGT9N5V53OJVV4KQNSO5AEMVJF66Q9ASUAAJG, Extended Request ID: not
 available
```

If you are interested in only the request ID use `<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />`.

Debug-level SDK logging

If you need more detail about what the SDK is doing, you can set the logging level of the `software.amazon.awssdk` logger to DEBUG. At this level, the SDK outputs a large amount of detail, so we recommend that you set this level to resolve errors using integration tests.

At this logging level, the SDK logs information about configuration, credentials resolution, execution interceptors, high-level TLS activity, request signing, and much more.

The following is a sampling of statements that are output by the SDK at DEBUG level for a `S3Client#listBuckets()` call.

```
DEBUG s.a.a.r.p.AwsRegionProviderChain:57 - Unable to load region from
 software.amazon.awssdk.regions.providers.SystemSettingsRegionProvider@324dcd31:Unable
```

```
 to load region from system settings. Region must be specified either via environment
 variable (AWS_REGION) or  system property (aws.region).
DEBUG s.a.a.c.i.h.l.ClasspathSdkHttpServiceProvider:85 - The HTTP implementation loaded
 is software.amazon.awssdk.http.apache.ApacheSdkHttpService@a23a01d
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Creating an interceptor
 chain that will apply interceptors in the following order:
 [software.amazon.awssdk.core.internal.interceptor.HttpChecksumValidationInterceptor@69b2f8e5,
 software.amazon.awssdk.awscore.interceptor.HelpfulUnknownHostExceptionInterceptor@6331250e,
 software.amazon.awssdk.awscore.eventstream.EventStreamInitialRequestInterceptor@a10c1b5,
 software.amazon.awssdk.awscore.interceptor.TraceIdExecutionInterceptor@644abb8f,
 software.amazon.awssdk.services.s3.auth.scheme.internal.S3AuthSchemeInterceptor@1a411233,
 software.amazon.awssdk.services.s3.endpoints.internal.S3ResolveEndpointInterceptor@70325d20,
 software.amazon.awssdk.services.s3.endpoints.internal.S3RequestSetEndpointInterceptor@7c2327fa
 software.amazon.awssdk.services.s3.internal.handlers.StreamingRequestInterceptor@4d847d32,
 software.amazon.awssdk.services.s3.internal.handlers.CreateBucketInterceptor@5f462e3b,
 software.amazon.awssdk.services.s3.internal.handlers.CreateMultipartUploadRequestInterceptor@3
 software.amazon.awssdk.services.s3.internal.handlers.DecodeUrlEncodedResponseInterceptor@58065
 software.amazon.awssdk.services.s3.internal.handlers.GetBucketPolicyInterceptor@3605c4d3,
 software.amazon.awssdk.services.s3.internal.handlers.S3ExpressChecksumInterceptor@585c13de,
 software.amazon.awssdk.services.s3.internal.handlers.AsyncChecksumValidationInterceptor@187eb9
 software.amazon.awssdk.services.s3.internal.handlers.SyncChecksumValidationInterceptor@726a6b9
 software.amazon.awssdk.services.s3.internal.handlers.EnableTrailingChecksumInterceptor@6ad11a5
 software.amazon.awssdk.services.s3.internal.handlers.ExceptionTranslationInterceptor@522b2631,
 software.amazon.awssdk.services.s3.internal.handlers.GetObjectInterceptor@3ff57625,
 software.amazon.awssdk.services.s3.internal.handlers.CopySourceInterceptor@1ee29c84,
 software.amazon.awssdk.services.s3.internal.handlers.ObjectMetadataInterceptor@7c8326a4]
DEBUG s.a.a.u.c.CachedSupplier:85 - (SsoOidcTokenProvider()) Cached value is stale and
 will be refreshed.
...
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Creating an interceptor
 chain that will apply interceptors in the following order:
 [software.amazon.awssdk.core.internal.interceptor.HttpChecksumValidationInterceptor@51351f28,
 software.amazon.awssdk.awscore.interceptor.HelpfulUnknownHostExceptionInterceptor@21618fa7,
 software.amazon.awssdk.awscore.eventstream.EventStreamInitialRequestInterceptor@15f2eda3,
 software.amazon.awssdk.awscore.interceptor.TraceIdExecutionInterceptor@34cf294c,
 software.amazon.awssdk.services.sso.auth.scheme.internal.SsoAuthSchemeInterceptor@4d7aaca2,
 software.amazon.awssdk.services.sso.endpoints.internal.SsoResolveEndpointInterceptor@604b1e1d,
 software.amazon.awssdk.services.sso.endpoints.internal.SsoRequestSetEndpointInterceptor@625668
...
DEBUG s.a.a.request:85 - Sending Request: DefaultSdkHttpFullRequest(httpMethod=GET,
 protocol=https, host=portal.sso.us-east-1.amazonaws.com, encodedPath=/federation/
credentials, headers=[amz-sdk-invocation-id, User-Agent, x-amz-sso_bearer_token],
 queryParameters=[role_name, account_id])
DEBUG s.a.a.c.i.h.p.s.SigningStage:85 - Using SelectedAuthScheme: smithy.api#noAuth
```

```
DEBUG s.a.a.h.a.i.c.SdkTlsSocketFactory:366 - Connecting socket to portal.sso.us-
east-1.amazonaws.com/18.235.195.183:443 with timeout 2000
...
DEBUG s.a.a.requestId:85 - Received successful response: 200, Request ID: bb4f40f4-
e920-4b5c-8648-58f26e7e08cd, Extended Request ID: not available
DEBUG s.a.a.request:85 - Received successful response: 200, Request ID: bb4f40f4-
e920-4b5c-8648-58f26e7e08cd, Extended Request ID: not available
DEBUG s.a.a.u.c.CachedSupplier:85 -
 (software.amazon.awssdk.services.sso.auth.SsoCredentialsProvider@b965857) Successfully
 refreshed cached value. Next Prefetch Time: 2024-04-25T22:03:10.097Z. Next Stale Time:
 2024-04-25T22:05:30Z
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Interceptor
 'software.amazon.awssdk.services.s3.endpoints.internal.S3RequestSetEndpointInterceptor@7c2327f
 modified the message with its modifyHttpRequest method.
...
DEBUG s.a.a.c.i.h.p.s.SigningStage:85 - Using SelectedAuthScheme: aws.auth#sigv4
...
DEBUG s.a.a.a.s.Aws4Signer:85 - AWS4 Canonical Request: GET
...
DEBUG s.a.a.h.a.a.i.s.DefaultV4RequestSigner:85 - AWS4 String to sign: AWS4-HMAC-SHA256
20240425T210631Z
20240425/us-east-1/s3/aws4_request
aafb7784627fa7a49584256cb746279751c48c2076f813259ef767ecce304d64
DEBUG s.a.a.h.a.i.c.SdkTlsSocketFactory:366 - Connecting socket to s3.us-
east-1.amazonaws.com/52.217.41.86:443 with timeout 2000
...
```

The following `log4j2.xml` file configures the previous output.

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

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

Enable wire logging

It can be useful to see the exact requests and responses that the SDK for Java 2.x sends and receives. If you need access to this information, you can temporarily enable it by adding the necessary configuration depending on the HTTP client the service client uses.

By default, synchronous service clients, such as the S3Client, use an underlying Apache HttpClient, and asynchronous service clients, such as the S3AsyncClient, use a Netty non-blocking HTTP client.

Here is a breakdown of HTTP clients you can use for the two categories of service clients:

Synchronous HTTP Clients	Asynchronous HTTP Clients
ApacheHttpClient (default)	NettyNioAsyncHttpClient (default)
UrlConnectionHttpClient	AwsCrtAsyncHttpClient

Consult the appropriate tab below for configuration settings you need to add depending on the underlying HTTP client.

> **Warning**
>
> We recommend you only use wire logging for debugging purposes. Disable it in your production environments because it can log sensitive data. It logs the full request or response without encryption, even for an HTTPS call. For large requests (e.g., to upload a file to Amazon S3) or responses, verbose wire logging can also significantly impact your application's performance.

ApacheHttpClient

Add the "org.apache.http.wire" logger to the `log4j2.xml` configuration file and set the level to "DEBUG".

The following `log4j2.xml` file turns on full wire logging for the Apache HttpClient.

```
<Configuration status="WARN">
 <Appenders>
```

```
  <Console name="ConsoleAppender" target="SYSTEM_OUT">
   <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
  </Console>
 </Appenders>

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

An additional Maven dependency on the `log4j-1.2-api` artifact is required for wire logging with Apache since it uses 1.2 under the hood.

The full set of Maven dependencies for log4j 2, including wire logging for the Apache HTTP client are shown in the following build file snippets.

Maven

```
...
<dependencyManagement>
    ...
    <dependencies>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-bom</artifactId>
            <version>VERSION</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
     </dependencies>
</dependencyManagement>
...
<!-- The following is needed for Log4j2 with SLF4J -->
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-slf4j2-impl</artifactId>
</dependency>
```

```
<!-- The following is needed for Apache HttpClient wire logging -->
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-1.2-api</artifactId>
</dependency>
...
```

Gradle–Kotlin DSL

```
...
dependencies {
    ...
    implementation(platform("org.apache.logging.log4j:log4j-bom:VERSION"))
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")
    implementation("org.apache.logging.log4j:log4j-1.2-api")
}
...
```

Use `2.20.0` for the minimum version of the `log4j-bom` artifact. For the latest version, use the version published to [Maven central](). Replace *`VERSION`* with version you'll use.

UrlConnectionHttpClient

To log details for service clients that use the `UrlConnectionHttpClient`, first create a `logging.properties` file with the following contents:

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

Set the following JVM system property with the full path of the `logging.properties`:

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

This configuration will log the only the headers of the request and response, for example:

```
<Request>  FINE: sun.net.www.MessageHeader@35a9782c11 pairs: {GET /fileuploadtest
 HTTP/1.1: null}{amz-sdk-invocation-id: 5f7e707e-4ac5-bef5-ba62-00d71034ffdc}
{amz-sdk-request: attempt=1; max=4}{Authorization: AWS4-HMAC-SHA256
 Credential=<deleted>/20220927/us-east-1/s3/aws4_request, SignedHeaders=amz-sdk-
```

```
invocation-id;amz-sdk-request;host;x-amz-content-sha256;x-amz-date;x-amz-te,
 Signature=e367fa0bc217a6a65675bb743e1280cf12fbe8d566196a816d948fdf0b42ca1a}{User-
Agent: aws-sdk-java/2.17.230 Mac_OS_X/12.5 OpenJDK_64-Bit_Server_VM/25.332-b08
 Java/1.8.0_332 vendor/Amazon.com_Inc. io/sync http/UrlConnection cfg/retry-mode/
legacy}{x-amz-content-sha256: UNSIGNED-PAYLOAD}{X-Amz-Date: 20220927T133955Z}{x-amz-
te: append-md5}{Host: tkhill-test1.s3.amazonaws.com}{Accept: text/html, image/gif,
 image/jpeg, *; q=.2, */*; q=.2}{Connection: keep-alive}
<Response> FINE: sun.net.www.MessageHeader@70a36a6611 pairs: {null: HTTP/1.1
 200 OK}{x-amz-id-2: sAFeZDOKdUMsBbkdjyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/
dFuOvr2tUb7Y1wEHGdJ3DSIxq0=}{x-amz-request-id: P9QW9SMZ97FKZ9X7}{Date: Tue,
 27 Sep 2022 13:39:57 GMT}{Last-Modified: Tue, 13 Sep 2022 14:38:12 GMT}{ETag:
 "2cbe5ad4a064cedec33b452bebf48032"}{x-amz-transfer-encoding: append-md5}{Accept-
Ranges: bytes}{Content-Type: text/plain}{Server: AmazonS3}{Content-Length: 67}
```

To see the request/response bodies, add `-Djavax.net.debug=all` to the JVM properties. This additional property logs a great deal of information, including all SSL information.

Within the log console or log file, search for "`GET`" or "`POST`" to quickly go to the section of the log containing actual requests and responses. Search for "`Plaintext before ENCRYPTION`" for requests and "`Plaintext after DECRYPTION`" for responses to see the full text of the headers and bodies.

NettyNioAsyncHttpClient

If your asynchronous service client uses the default `NettyNioAsyncHttpClient`, add two additional loggers to your `log4j2.xml` file to log HTTP headers and request/response bodies.

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

Here is a complete `log4j2.xml` example:

```
<Configuration status="WARN">
    <Appenders>
        <Console name="ConsoleAppender" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m
%n" />
        </Console>
    </Appenders>

    <Loggers>
        <Root level="WARN">
```

```
            <AppenderRef ref="ConsoleAppender"/>
        </Root>
        <Logger name="software.amazon.awssdk" level="WARN" />
        <Logger name="software.amazon.awssdk.request" level="DEBUG" />
        <Logger name="io.netty.handler.logging" level="DEBUG" />
        <Logger name="io.netty.handler.codec.http2.Http2FrameLogger" level="DEBUG" /
>
    </Loggers>
</Configuration>
```

These settings log all header details and request/response bodies.

AwsCrtAsyncHttpClient

If you have configured your service client to use an instance of `AwsCrtAsyncHttpClient`, you can log details by setting JVM system properties or programmatically.

<table>
<tr><th colspan="2">Log to a file at "Debug" level</th></tr>
<tr><td>Using system properties:
<pre>-Daws.crt.log.level=Trace
-Daws.crt.log.destination=File
-Daws.crt.log.filename=<path to file></pre></td><td>Programmatically:
<pre>import software.amazon.awssdk.crt.Log;

// Execute this statement before constructing the
 SDK service client.
Log.initLoggingToFile(Log.LogLevel.Trace,
 "<path to file>");</pre></td></tr>
<tr><th colspan="2">Log to the console at "Debug" level</th></tr>
<tr><td>Using system properties:
<pre>-Daws.crt.log.level=Trace
-Daws.crt.log.destination=Stdout</pre></td><td>Programmatically:
<pre>import software.amazon.awssdk.crt.Log;

// Execute this statement before constructing the
 SDK service client.
Log.initLoggingToStdout(Log.LogLevel.Trace);</pre></td></tr>
</table>

For security reasons, at the "Trace" level the `AwsCrtAsyncHttpClient` logs only response headers. Request headers, request bodies, and response bodies are not logged.

Set the JVM TTL for DNS name lookups

The Java virtual machine (JVM) caches DNS name lookups. When the JVM resolves a hostname to an IP address, it caches the IP address for a specified period of time, known as the *time-to-live* (TTL).

Because AWS resources use DNS name entries that occasionally change, we recommend that you configure your JVM with a TTL value of 5 seconds. This ensures that when a resource's IP address changes, your application will be able to receive and use the resource's new IP address by requerying the DNS.

On some Java configurations, the JVM default TTL is set so that it will *never* refresh DNS entries until the JVM is restarted. Thus, if the IP address for an AWS resource changes while your application is still running, it won't be able to use that resource until you *manually restart* the JVM and the cached IP information is refreshed. In this case, it's crucial to set the JVM's TTL so that it will periodically refresh its cached IP information.

How to set the JVM TTL

To modify the JVM's TTL, set the [networkaddress.cache.ttl] security property value, set the `networkaddress.cache.ttl` property in the `$JAVA_HOME/jre/lib/security/java.security` file for Java 8 or `$JAVA_HOME/conf/security/java.security` file for Java 11 or higher.

The following is a snippet from a `java.security` file that shows the TTL cache set to 5 seconds.

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

All applications that run on the JVM represented by the`$JAVA_HOME` environment variable use this setting.

Best practices for AWS SDK for Java 2.x

This section lists best practices for using the SDK for Java 2.x.

Topics

Reuse an SDK client, if possible

Each SDK client maintains its own HTTP connection pool. A connection that already exists in the pool can be reused by a new request to cut down the time to establish a new connection. We recommend sharing a single instance of the client to avoid the overhead of having too many connection pools that aren't used effectively. All SDK clients are thread safe.

If you don't want to share a client instance, call `close()` on the instance to release the resources when the client is not needed.

Close input streams from client operations

For streaming operations such as `S3Client#getObject`, if you are working with `ResponseInputStream` directly, we recommend that you do the following:

- Read all the data from the input stream as soon as possible.
- Close the input stream as soon as possible.

We make these recommendations because the input stream is a direct stream of data from the HTTP connection and the underlying HTTP connection can't be reused until all data from the stream has been read and the stream is closed. If these rules are not followed, the client can run out of resources by allocating too many open, but unused, HTTP connections.

Tune HTTP configurations based on performance tests

The SDK provides a set of default http configurations that apply to general use cases. We recommend that customers tune HTTP configurations for their applications based on their use cases.

As a good starting point, the SDK offers a smart configuration defaults feature. This feature is available starting with version 2.17.102. You choose a mode depending on your use case, which provides sensible configuration values.

Use OpenSSL for the Netty-based HTTP client

By default, the SDK's `NettyNioAsyncHttpClient` uses the JDK's default SSL implementation as the `SslProvider`. Our testing found that OpenSSL performs better than JDK's default implementation. The Netty community also recommends using OpenSSL.

To use OpenSSL, add `netty-tcnative` to your dependencies. For configuration details, see the Netty project documentation.

After you have `netty-tcnative` configured for your project, the `NettyNioAsyncHttpClient` instance will automatically select OpenSSL. Alternatively, you can set the `SslProvider` explicitly using the `NettyNioAsyncHttpClient` builder as shown in the following snippet.

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

Configure API timeouts

The SDK provides default values for some timeout options, such as connection timeout and socket timeouts, but not for API call timeouts or individual API call attempt timeouts. It is a good practice to set timeouts for both the individual attempts and the entire request. This will ensure your application fails fast in an optimal way when there are transient issues that could cause request attempts to take longer to complete or fatal network issues.

You can configure timeouts for all requests made by a service clients using `ClientOverrideConfiguration#apiCallAttemptTimeout` and `ClientOverrideConfiguration#apiCallTimeout`.

The following example shows the configuration of an Amazon S3 client with custom timeout values.

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

`apiCallAttemptTimeout`

This setting sets the amount of time for a single HTTP attempt, after which the API call can be retried.

`apiCallTimeout`

The value for this property configures the amount of time for the entire execution, including all retry attempts.

As an alternative to setting these timeout values on the service client, you can use `RequestOverrideConfiguration#apiCallTimeout()` and `RequestOverrideConfiguration#apiCallAttemptTimeout()` to configure a single request .

The following example configures a single `listBuckets` request with custom timeout values.

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

When you use these properties together, you set a hard limit on the total time spent on all attempts across retries. You also set an individual HTTP request to fail fast on a slow request.

Use metrics

The SDK for Java can collect metrics for the service clients in your application. You can use these metrics to identify performance issues, review overall usage trends, review service client exceptions returned, or to dig in to understand a particular issue.

We recommend that you collect metrics, then analyze the Amazon CloudWatch Logs, in order to gain a deeper understanding of your application's performance.

Troubleshooting FAQs

As you use the AWS SDK for Java 2.x in your applications, you might encounter the runtime errors listed in this topic. Use the suggestions here to help you uncover the root cause and resolve the error.

How do I fix "`java.net.SocketException`: Connection reset" or "server failed to complete the response" error?

A connection reset error indicates that your host, the AWS service, or any intermediary party (for example, a NAT gateway, a proxy, a load balancer) closed the connection before the request was complete. Because there are many causes, finding a solution requires that you know why the connection is being closed. The following items commonly cause a connection to be closed.

- **The connection is inactive.** This is common for streaming operations, where data is not being written to or from the wire for a period of time, so an intermediary party detects the connection as dead and closes it. To prevent this, be sure your application actively downloads or uploads data.
- **You've closed the HTTP or SDK client.** Be sure not to close resources while they are in use.
- **A misconfigured proxy.** Try to bypass any proxies that you've configured to see if it fixes the problem. If this fixes the issue, the proxy is closing your connection for some reason. Research your specific proxy to determine why it's closing the connection.

If you cannot identify the problem, try running a TCP dump for an affected connection at the client edge of your network (for example, after any proxies that you control).

If you see that the AWS endpoint is sending a `TCP RST` (reset), contact the affected service to see if they can determine why the reset is occurring. Be prepared to provide request IDs and timestamps of when the issue occurred. The AWS support team might also benefit from wire logs that show exactly what bytes your application is sending and receiving and when.

How do I fix "connection timeout"?

A connection timeout error indicates that your host, the AWS service, or any intermediary party (for example, a NAT gateway, a proxy, a load balancer) failed to establish a new connection with the server within the configured connection timeout. The following items describe common causes of this issue.

- **The configured connection timeout is too low.** By default, the connection timeout is 2 seconds in the AWS SDK for Java 2.x. If you set the connection timeout too low, you may get this error. The recommended connection timeout is 1 second if you make only in-region calls and 3 seconds if you make cross-region requests.
- **A misconfigured proxy.** Try to bypass any proxies that you configured to see if it fixes the problem. If this fixes the issue, the proxy is the reason for the connection timeout. Research your specific proxy to determine why that is happening

If you cannot identify the problem, try running a TCP dump for an affected connection at the client edge of your network (for example, after any proxies that you control) to investigate any network issue.

How do I fix "`java.net.SocketTimeoutException`: Read timed out"?

A read timed out error indicates that the JVM attempted to read data from the underlying operating system, but data was not returned within the time configured via the SDK. This error can occur if the operating system, the AWS service, or any intermediary party (for example, a NAT gateway, a proxy, a load balancers) fails to send data within the time expected by the JVM. Because there are many causes, finding a solution requires that you know why the data is not being returned.

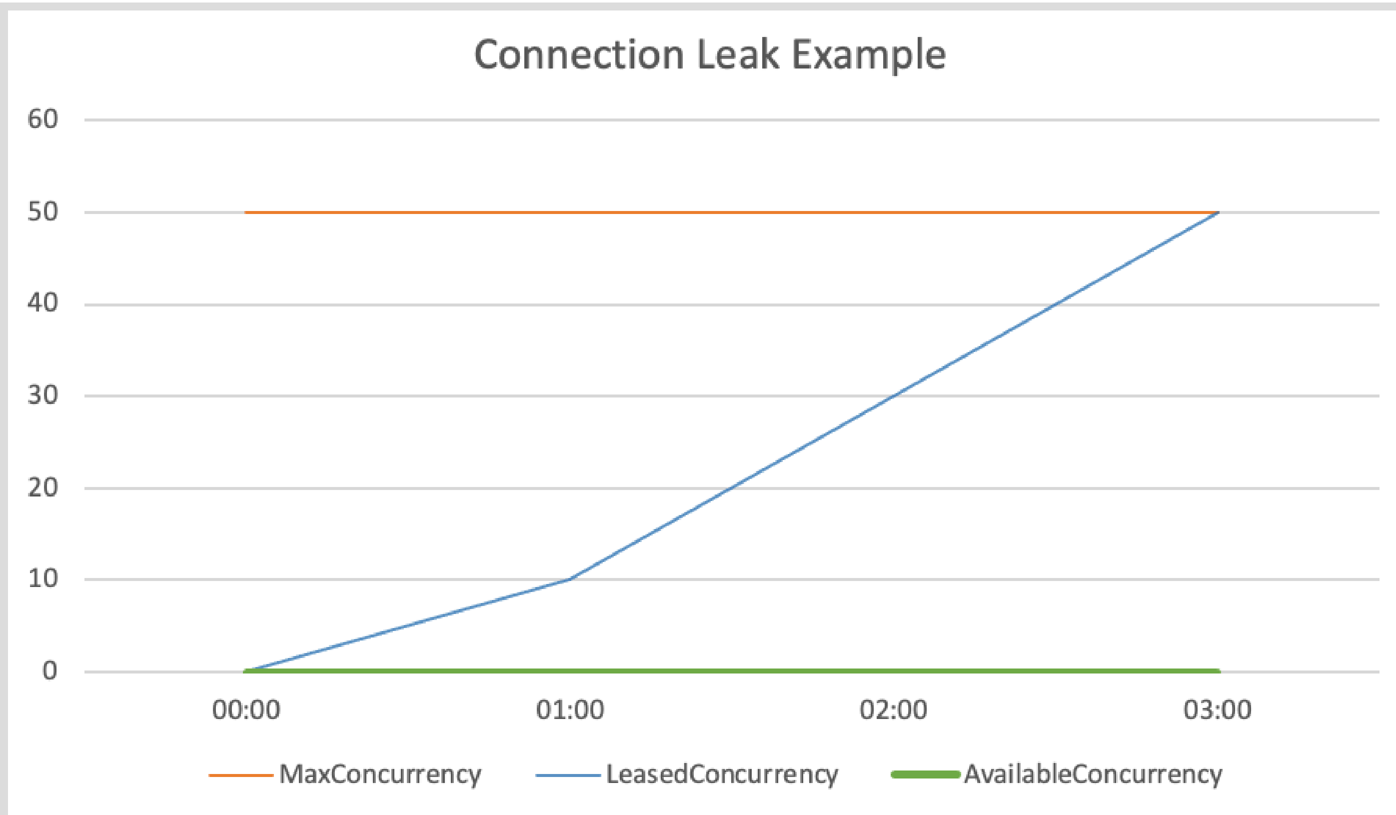
Try running a TCP dump for an affected connection at the client edge of your network (for example, after any proxies that you control).

If you see that the AWS endpoint is sending a `TCP RST` (reset), contact the affected service. Be prepared to provide request IDs and timestamps of when the issue occurred. The AWS support team might also benefit from wire logs that show exactly what bytes your application is sending and receiving and when.

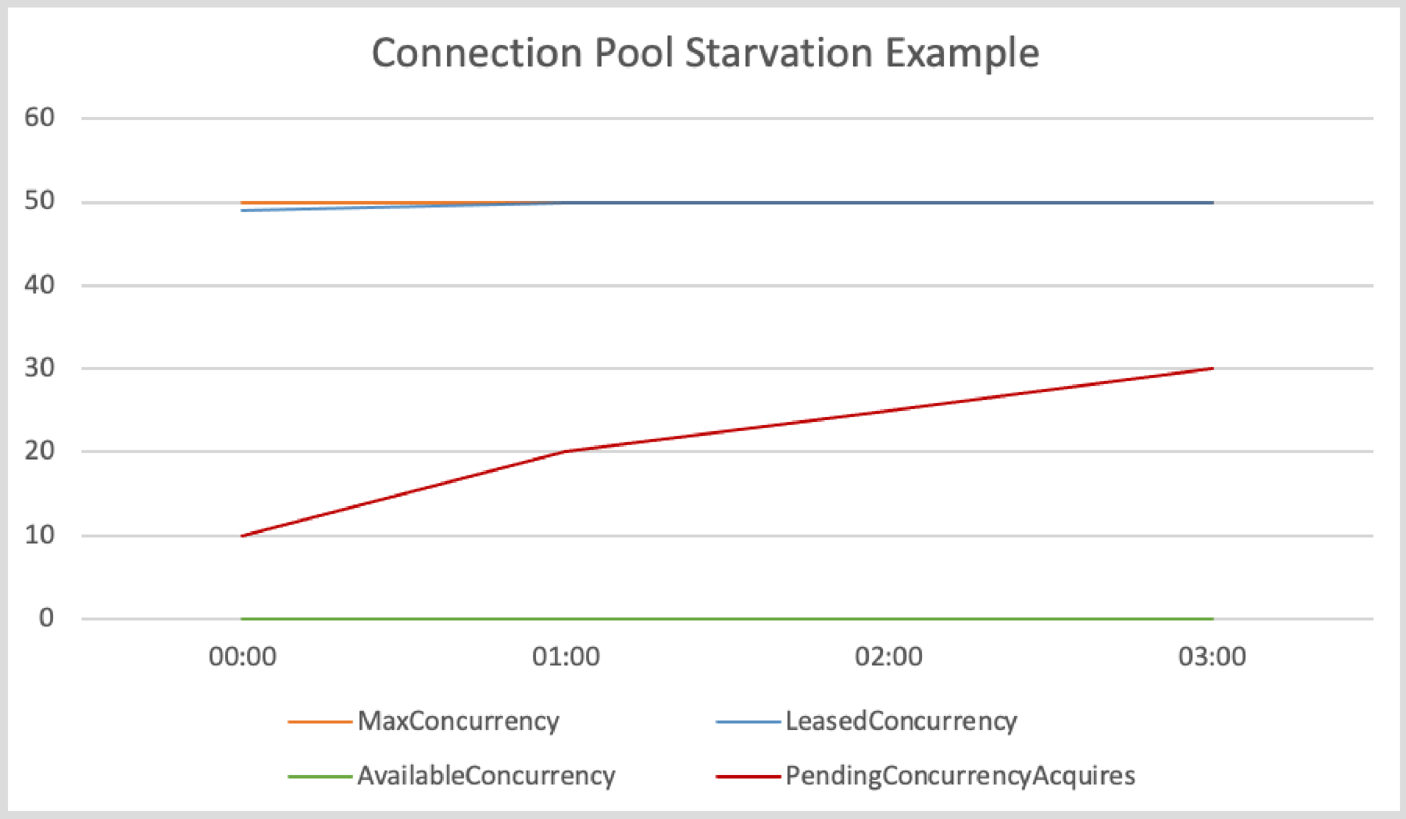
How do I fix "Unable to execute HTTP request: Timeout waiting for connection from pool" error?

This error indicates that a request cannot get a connection from the pool within the specified maximum time. To troubleshoot the issue, we recommend that you enable SDK client-side metrics to publish metrics to Amazon CloudWatch. The HTTP metrics can help narrow down the root cause. The following items describe common causes of this error.

- **Connection leak.** You can investigate this by checking `LeasedConcurrency`, `AvailableConcurrency`, and `MaxConcurrency` metrics. If `LeasedConcurrency` increases until it reaches `MaxConcurrency` but never decreases, there may be a connection leak. A common cause of a leak is because a streaming operation—such as a S3 `getObject` method—is not closed. We recommend that your application read all data from the input stream as soon as possible and close the input stream afterwards. The following chart shows what SDK metrics might look like for connection leak.



- **Connection pool starvation.** This can happen if your request rate is too high and the connection pool size that has been configured cannot meet the request demand. The default connection pool size is 50, and when the connections in the pool reach the maximum, the HTTP client queues incoming requests until connections become available. The following chart shows what SDK metrics might look like for connection pool starvation.



To mitigate this issue, consider taking any of the following actions.

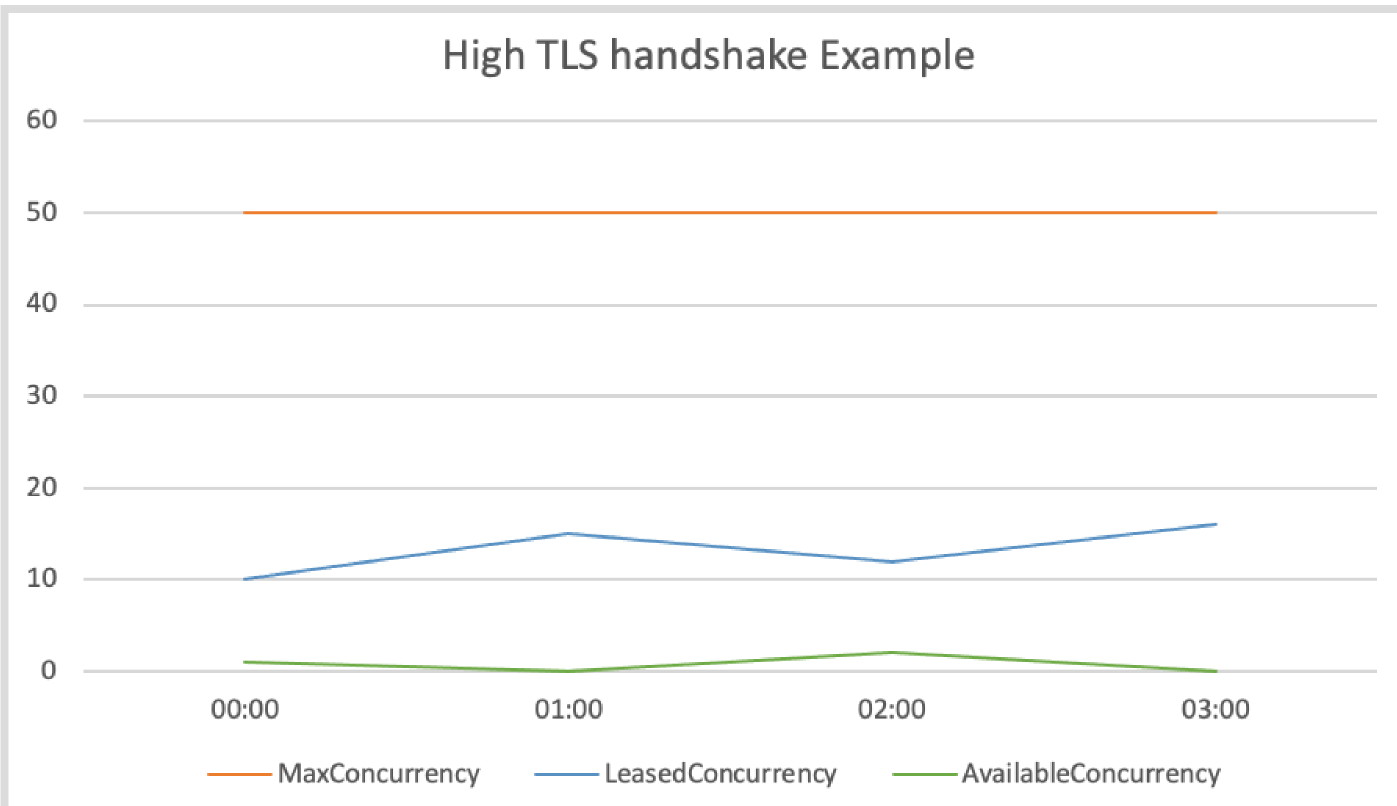
- Increase the connection pool size,
- Increase acquire timeout.
- Slow the request rate.

By increasing the maximum number of connections, client throughput can increase (unless the network interface is already fully utilized). However, you can eventually hit operation system limitations on the number of file descriptors used by the process. If you already fully use your network interface or cannot further increase your connection count, try increasing the acquire timeout. With the increase, you gain extra time for requests to acquire a connection before timing out. If the connections don't free up, the subsequent requests will still timeout.

If you are unable to fix the issue by using the first two mechanisms, slow the request rate by trying the following options.

- Smooth out your requests so that large traffic bursts don't overload the client.
- Be more efficient with calls to AWS services.
- Increase the number of hosts sending requests.

- **I/O Threads are too busy.** This only applies if you are using an asynchronous SDK client with `NettyNioAsyncHttpClient`. If the `AvailableConcurrency` metric is not low—indicating that connections are available in the pool—but `ConcurrencyAcquireDuration` is high, it might be because I/O threads are not able to handle the requests. Be sure you are not passing `Runnable:run` as a future completion executor and performing time-consuming task in the response future completion chain since this can block an I/O thread. If that is not the case, consider increasing the number of I/O threads by using the `eventLoopGroupBuilder` method. For reference, the default number of I/O threads for a `NettyNioAsyncHttpClient` instance is twice the number of CPU cores of the host.
- **High TLS handshake latency.** If your `AvailableConcurrency` metric is near 0 and `LeasedConcurrency` is lower than `MaxConcurrency`, it might be because the TLS handshake latency is high. The following chart shows what SDK metrics might look like for high TLS handshake latency.



For HTTP clients offered by the Java SDK that are not based on CRT, try enabling TLS logs to troubleshoot TLS issues. For the AWS CRT-based HTTP client, try enabling AWS CRT logs. If you see that the AWS endpoint seems to take a long time to perform a TLS handshake, you should contact the affected service.

How do I fix a `NoClassDefFoundError`, `NoSuchMethodError` or `NoSuchFieldError`?

A `NoClassDefFoundError` indicates that a class could not be loaded at runtime. The two most common causes for this error are:

- the class does not exist in the classpath because the JAR is missing or the wrong version of the JAR is on the classpath.
- the class failed to load because its static initializer threw an exception.

Similarly, `NoSuchMethodErrors` and `NoSuchFieldErrors` typically result from a mismatched JAR version. We recommend that you perform the following steps.

1. **Check your dependencies** to make sure that you're using the *same version of all SDK jars*. The most common reason that a class, method, or field cannot be found is when you upgrade to a new client version but you continue to use an old 'shared' SDK dependency version. The new client version might attempt to use classes that exist only in newer 'shared' SDK dependencies. Try running `mvn dependency:tree` or `gradle dependencies` (for Gradle) to verify that the SDK library versions all match. To avoid this issue completely in the future, we recommend using [BOM (Bill of Materials)](#) to manage SDK module versions.

   The following example shows you an example of mixed SDK versions.

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

   The version of `dynamodb` is 2.20.00 and the version of `aws-core` is 2.13.19. The `aws-core` artifact version should also be 2.20.00.

2. **Check statements early in your logs** to see if a class is failing to load because of a static initialization failure. The first time the class fails to load, it may throw a different, more useful exception that specifies *why* the class cannot be loaded. This potentially useful exception occurs only once, so later log statements will only report that the class is not found.

3. **Check your deployment process** to make sure that it actually deploys required JAR files along with your application. It's possible that you're building with the correct version, but the process that creates the classpath for your application is excluding a required dependency.

How do I fix a "`SignatureDoesNotMatch`" error or "The request signature we calculated does not match the signature you provided" error?

A `SignatureDoesNotMatch` error indicates that the signature generated by the AWS SDK for Java and the signature generated by the AWS service do not match. The following items describe potential causes.

- A proxy or intermediary party modifies the request. For example, a proxy or load balancer might modify a header, path or query string that was signed by the SDK.
- The service and SDK differ in the way they encode the request when each generates the string to sign.

To debug this issue, we recommend that you enable debug logging for the SDK. Try to reproduce the error and find the canonical request that the SDK generated. In the log, the canonical request is labeled with `AWS4 Canonical Request: ...` and the string to sign is labeled `AWS4 String to sign: ...`.

If you cannot enable debugging—for example, because it's only reproducible in production—add logic to your application that logs information about the request when the error occurs. You can then use that information to try to replicate the error outside of production in an integration test with debug logging enabled.

After you have collected the canonical request and string to sign, compare them against the AWS Signature Version 4 specification to determine if there are any issues in the way the SDK generated the string to sign. If something seems wrong, you can create a GitHub bug report to the AWS SDK for Java.

If nothing appears wrong, you can compare the SDK's string to sign with the string to sign that some AWS services return as part of the failure response (Amazon S3, for example) . If this isn't available, you should contact the affected service to see what canonical request and string to sign they generated for comparison. These comparisons can help to identify intermediary parties that might have modified the request or encoding differences between the service and client.

For more background information about signing requests, see Signing AWS API requests in the AWS Identity and Access Management User Guide.

Example of a canonical request

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

Example of a string to sign

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

How do I fix "`java.lang.IllegalStateException`: Connection pool shut down" error?

This error indicates the underlying Apache HTTP connection pool was closed. The following items describe potential causes.

- **The SDK client was closed prematurely.** The SDK only closes the connection pool when the associated client is closed. Be sure not to close resources while they are in use.
- **A `java.lang.Error` was thrown.** Errors such as `OutOfMemoryError` cause an Apache HTTP connection pool to shut down. Examine your logs for error stack traces. Also review your code for places where it catches `Throwables` or `Errors` but swallows the output that prevents the error from surfacing. If your code does not report errors, rewrite the code so information is logged. The logged information helps determine the root cause of the error.
- **You attempted to use the credentials provider returned from `DefaultCredentialsProvider#create()` after it was closed**.

`DefaultCredentialsProvider#create` returns a singleton instance, so if it's closed and your code calls the `resolveCredentials` method, the exception is thrown after cached credentials (or token) expire.

Check your code for places where the `DefaultCredentialsProvider` is closed, as shown in the following examples.

- The singleton instance is closed by calling `DefaultCredentialsProvider#close()`.

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

// Make calls to AWS services.

defaultCredentialsProvider.close();  // Explicit close.

// Make calls to AWS services.

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

- Invoke `DefaultCredentialsProvider#create()` in a try-with-resources block.

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

    // Make calls to AWS services.

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

// Make calls to AWS services.

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

Create a new, non-singleton instance by calling `DefaultCredentialsProvider.builder().build()` if your code has closed the singleton instance and you need to resolve credentials by using a `DefaultCredentialsProvider`.

Use features of the AWS SDK for Java 2.x

General features

The SDK for Java 2.x contains several features that make programming against AWS services easier.

- The SDK hides the complex mechanisms behind retrieving paginated results and polling for resources.
- Asynchronous programming with non-blocking I/O helps you write concurrent code with better performance. The SDK provides the benefits of HTTP/2, such as reduced latency, where possible.
- The Java SDK can generate metrics to help you monitor the operational health of your applications.

Service-specific features

In addition to the general features mentioned previously, the Java SDK provides features for specific AWS services.

- **Amazon S3** – To simplify your work with files and directories with Amazon S3, the SDK provides the S3 Transfer Manager. To improve performance and reliability while using the SDK's standard asynchronous S3 API, the SDK offers the AWS CRT-based S3 client.
- **DynamoDB –** Object-oriented, mapping capability is provided by the DynamoDB Enhanced Client API. Work with JSON-style, document-oriented data by using the Enhanced Document API.
- **IAM –** The IAM Policy Builder API provides a type-safe, object-oriented way to create IAM policies.

Work with paginated results using the AWS SDK for Java 2.x

Many AWS operations return paginated results when the response object is too large to return in a single response. In the AWS SDK for Java 1.0, the response contains a token you use to retrieve the next page of results. In contrast, the AWS SDK for Java 2.x has autopagination methods that make multiple service calls to get the next page of results for you automatically. You only have to write code that processes the results. Autopagination is available for both synchronous and asynchronous clients.

Note

These code snippets assume that you understand the basics of using the SDK, and have configured your environment with single sign-on access.

Synchronous pagination

The following examples demonstrate synchronous pagination methods to list objects in an Amazon S3 bucket.

Iterate over pages

The first example demonstrates the use of a `listRes` paginator object, a `ListObjectsV2Iterable` instance, to iterate through all the response pages with the `stream` method. The code streams over the response pages, converts the response stream to a stream of `S3Object` content, and then processes the content of the Amazon S3 object.

The following imports apply to all examples in this synchronous pagination section.

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
```

```
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

```
                ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                                .bucket(bucketName)
                                .maxKeys(1)
                                .build();

                ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
                // Process response pages
                listRes.stream()
                                .flatMap(r -> r.contents().stream())
                                .forEach(content -> System.out
                                                .println(" Key: " + content.key() + "
 size = " + content.size()));
```

See the complete example on GitHub.

Iterate over objects

The following examples show ways to iterate over the objects returned in the response instead of the pages of the response. The `contents` method of `ListObjectsV2Iterable` class returns an `SdkIterable` that provides several methods to process the underlying content elements.

Use a stream

The following snippet uses the `stream` method on the response content to iterate over the paginated item collection.

```
                // Helper method to work with paginated collection of items directly.
                listRes.contents().stream()
                                .forEach(content -> System.out
                                                .println(" Key: " + content.key() + "
 size = " + content.size()));
```

See the complete example on GitHub.

Use a for-each loop

Since `SdkIterable` extends the `Iterable` interface, you can process the contents like any `Iterable`. The following snippet uses standard `for-each` loop to iterate through the contents of the response.

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

See the complete example on GitHub.

Manual pagination

If your use case requires it, manual pagination is still available. Use the next token in the response object for the subsequent requests. The following example uses a `while` loop.

```
                ListObjectsV2Request listObjectsReqManual =
ListObjectsV2Request.builder()
                                .bucket(bucketName)
                                .maxKeys(1)
                                .build();

                boolean done = false;
                while (!done) {
                        ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
                        for (S3Object content : listObjResponse.contents()) {
                                System.out.println(content.key());
                        }

                        if (listObjResponse.nextContinuationToken() == null) {
                                done = true;
                        }

                        listObjectsReqManual = listObjectsReqManual.toBuilder()

.continuationToken(listObjResponse.nextContinuationToken())
                                        .build();
```

```
                }
```

See the complete example on GitHub.

Asynchronous pagination

The following examples demonstrate asynchronous pagination methods to list DynamoDB tables.

Iterate over pages of table names

The following two examples use an asynchronous DynamoDB client that call the `listTablesPaginator` method with a request to get a `ListTablesPublisher`. `ListTablesPublisher` implements two interfaces, which provides many options to process responses. We'll look at methods of each interface.

Use a `Subscriber`

The following code example demonstrates how to process paginated results by using the `org.reactivestreams.Publisher` interface implemented by `ListTablesPublisher`. To learn more about the reactive streams model, see the Reactive Streams GitHub repo.

The following imports apply to all examples in this asynchronous pagination section.

Imports

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

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

The following code acquires a `ListTablesPublisher` instance.

```
        // Creates a default client with credentials and region loaded from the
```

```
        // environment.
        final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

        ListTablesRequest listTablesRequest =
ListTablesRequest.builder().limit(3).build();
        ListTablesPublisher publisher =
asyncClient.listTablesPaginator(listTablesRequest);
```

The following code uses an anonymous implementation of `org.reactivestreams.Subscriber` to process the results for each page.

The `onSubscribe` method calls the `Subscription.request` method to initiate requests for data from the publisher. This method must be called to start getting data from the publisher.

The subscriber's `onNext` method processes a response page by accessing all the table names and printing out each one. After the page is processed, another page is requested from the publisher. This method that is called repeatedly until all pages are retrieved.

The `onError` method is triggered if an error occurs while retrieving data. Finally, the `onComplete` method is called when all pages have been requested.

```
        // A Subscription represents a one-to-one life-cycle of a Subscriber
subscribing
        // to a Publisher.
        publisher.subscribe(new Subscriber<ListTablesResponse>() {
            // Maintain a reference to the subscription object, which is required to
request
            // data from the publisher.
            private Subscription subscription;

            @Override
            public void onSubscribe(Subscription s) {
                subscription = s;
                // Request method should be called to demand data. Here we request a
single
                // page.
                subscription.request(1);
            }

            @Override
            public void onNext(ListTablesResponse response) {
                response.tableNames().forEach(System.out::println);
```

```
                // After you process the current page, call the request method to 
 signal that
                // you are ready for next page.
                subscription.request(1);
            }

            @Override
            public void onError(Throwable t) {
                // Called when an error has occurred while processing the requests.
            }

            @Override
            public void onComplete() {
                // This indicates all the results are delivered and there are no more 
 pages
                // left.
            }
        });
```

See the complete example on GitHub.

Use a `Consumer`

The `SdkPublisher` interface that `ListTablesPublisher` implements has a `subscribe` method that takes a `Consumer` and returns a `CompletableFuture<Void>`.

The `subscribe` method from this interface can be used for simple use cases when an `org.reactivestreams.Subscriber` might be too much overhead. As the code below consumes each page, it calls the `tableNames` method on each. The `tableNames` method returns a `java.util.List` of DynamoDB table names that are processed with the `forEach` method.

```
        // Use a Consumer for simple use cases.
        CompletableFuture<Void> future = publisher.subscribe(
                response -> response.tableNames()
                        .forEach(System.out::println));
```

See the complete example on GitHub.

Iterate over table names

The following examples show ways to iterate over the objects returned in the response instead of the pages of the response. Similar to the synchronous Amazon S3 example previously shown with its `contents` method, the DynamoDB asynchronous result class, `ListTablesPublisher` has the `tableNames` convenience method to interact with the underlying item collection. The return type of the `tableNames` method is an [SdkPublisher](#) that can be used to request items across all pages.

Use a Subscriber

The following code acquires an `SdkPublisher` of the underlying collection of table names.

```
        // Create a default client with credentials and region loaded from the
        // environment.
        final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

        ListTablesRequest listTablesRequest =
 ListTablesRequest.builder().limit(3).build();
        ListTablesPublisher listTablesPublisher =
 asyncClient.listTablesPaginator(listTablesRequest);
        SdkPublisher<String> publisher = listTablesPublisher.tableNames();
```

The following code uses an anonymous implementation of `org.reactivestreams.Subscriber` to process the results for each page.

The subscriber's `onNext` method processes an individual element of the collection. In this case, it's a table name. After the table name is processed, another table name is requested from the publisher. This method that is called repeatedly until all table names are retrieved.

```
        // Use a Subscriber.
        publisher.subscribe(new Subscriber<String>() {
            private Subscription subscription;

            @Override
            public void onSubscribe(Subscription s) {
                subscription = s;
                subscription.request(1);
            }

            @Override
            public void onNext(String tableName) {
                System.out.println(tableName);
                subscription.request(1);
            }
```

```
            @Override
            public void onError(Throwable t) {
            }

            @Override
            public void onComplete() {
            }
        });
```

See the complete example on GitHub.

Use a Consumer

The following example uses the `subscribe` method of `SdkPublisher` that takes a `Consumer` to process each item.

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

See the complete example on GitHub.

Use third-party library

You can use other third party libraries instead of implementing a custom subscriber. This example demonstrates the use of RxJava, but any library that implements the reactive stream interfaces can be used. See the RxJava wiki page on GitHub for more information on that library.

To use the library, add it as a dependency. If using Maven, the example shows the POM snippet to use.

POM Entry

```
<dependency>
      <groupId>io.reactivex.rxjava3</groupId>
      <artifactId>rxjava</artifactId>
      <version>3.1.6</version>
</dependency>
```

Code

```
        DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();
        ListTablesPublisher publisher =
asyncClient.listTablesPaginator(ListTablesRequest.builder()
                .build());

        // The Flowable class has many helper methods that work with
        // an implementation of an org.reactivestreams.Publisher.
        List<String> tables = Flowable.fromPublisher(publisher)
                .flatMapIterable(ListTablesResponse::tableNames)
                .toList()
                .blockingGet();
        System.out.println(tables);
```

See the complete example on GitHub.

Poll for resource states in the AWS SDK for Java 2.x: Waiters

The waiters utility of the AWS SDK for Java 2.x enables you to validate that AWS resources are in a specified state before performing operations on those resources.

A *waiter* is an abstraction used to poll AWS resources, such as DynamoDB tables or Amazon S3 buckets, until a desired state is reached (or until a determination is made that the resource won't ever reach the desired state). Instead of writing logic to continuously poll your AWS resources, which can be cumbersome and error-prone, you can use waiters to poll a resource and have your code continue to run after the resource is ready.

Prerequisites

Before you can use waiters in a project with the AWS SDK for Java, you must complete the steps in Setting up the AWS SDK for Java 2.x.

You must also configure your project dependencies (for example, in your `pom.xml` or `build.gradle` file) to use version `2.15.0` or later of the AWS SDK for Java.

For example:

```
<project>
  <dependencyManagement>
   <dependencies>
```

```
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.15.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
</project>
```

Using waiters

To instantiate a waiters object, first create a service client. Set the service client's `waiter()` method as the value of the waiter object. Once the waiter instance exists, set its response options to execute the appropriate code.

Synchronous programming

The following code snippet shows how to wait for a DynamoDB table to exist and be in an **ACTIVE** state.

```
DynamoDbClient dynamo = DynamoDbClient.create();
DynamoDbWaiter waiter = dynamo.waiter();

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

// print out the matched response with a tableStatus of ACTIVE
waiterResponse.matched().response().ifPresent(System.out::println);
```

Asynchronous programming

The following code snippet shows how to wait for a DynamoDB table to no longer exist.

```
DynamoDbAsyncClient asyncDynamo = DynamoDbAsyncClient.create();
DynamoDbAsyncWaiter asyncWaiter = asyncDynamo.waiter();

CompletableFuture<WaiterResponse<DescribeTableResponse>> waiterResponse =
          asyncWaiter.waitUntilTableNotExists(r -> r.tableName("myTable"));
```

```
waiterResponse.whenComplete((r, t) -> {
  if (t == null) {
   // print out the matched ResourceNotFoundException
   r.matched().exception().ifPresent(System.out::println);
  }
}).join();
```

Configure waiters

You can customize the configuration for a waiter by using the `overrideConfiguration()` on its builder. For some operations, you can apply a custom configuration when you make the request.

Configure a waiter

The following code snippet shows how to override the configuration on a waiter.

```
// sync
DynamoDbWaiter waiter =
   DynamoDbWaiter.builder()
          .overrideConfiguration(b -> b.maxAttempts(10))
          .client(dynamoDbClient)
          .build();
// async
DynamoDbAsyncWaiter asyncWaiter =
   DynamoDbAsyncWaiter.builder()
          .client(dynamoDbAsyncClient)
          .overrideConfiguration(o -> o.backoffStrategy(
               FixedDelayBackoffStrategy.create(Duration.ofSeconds(2))))
          .scheduledExecutorService(Executors.newScheduledThreadPool(3))
          .build();
```

Override configuration for a specific request

The following code snippet shows how to override the configuration for a waiter on a per-request basis. Note that only some operations have customizable configurations.

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

asyncWaiter.waitUntilTableExists(b -> b.tableName("myTable"),
                 o -> o.waitTimeout(Duration.ofMinutes(1)));
```

Code examples

For a complete example using waiters with DynamoDB, see [CreateTable.java] in the AWS Code Examples Repository.

For a complete example using waiters with Amazon S3, see [S3BucketOps.java] in the AWS Code Examples Repository.

Use asynchronous programming

The AWS SDK for Java 2.x features asynchronous clients with non-blocking I/O support that implement high concurrency across a few threads. However, total non-blocking I/O is not guaranteed. Asynchronous client may perform blocking calls in some cases such as credential retrieval, request signing using AWS Signature Version 4 (SigV4), or endpoint discovery.

Synchronous methods block your thread's execution until the client receives a response from the service. Asynchronous methods return immediately, giving control back to the calling thread without waiting for a response.

Because an asynchronous method returns before a response is available, you need a way to get the response when it's ready. The methods for asynchronous client in 2.x of the AWS SDK for Java return *CompletableFuture objects* that allow you to access the response when it's ready.

Non-streaming operations

For non-streaming operations, asynchronous method calls are similar to synchronous methods. However, the asynchronous methods in the AWS SDK for Java return a CompletableFuture object that contains the results of the asynchronous operation *in the future*.

Call the `CompletableFuture` `whenComplete()` method with an action to complete when the result is available. `CompletableFuture` implements the `Future` interface, so you can also get the response object by calling the `get()` method.

The following is an example of an asynchronous operation that calls a Amazon DynamoDB function to get a list of tables, receiving a `CompletableFuture` that can hold a ListTablesResponse object. The action defined in the call to `whenComplete()` is done only when the asynchronous call is complete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

Code

```
public class DynamoDBAsyncListTables {

    public static void main(String[] args) throws InterruptedException {

        // Create the DynamoDbAsyncClient object
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient client = DynamoDbAsyncClient.builder()
                .region(region)
                .build();

        listTables(client);
    }

    public static void listTables(DynamoDbAsyncClient client) {

        CompletableFuture<ListTablesResponse> response =
 client.listTables(ListTablesRequest.builder()
                .build());

        // Map the response to another CompletableFuture containing just the table
 names
        CompletableFuture<List<String>> tableNames =
 response.thenApply(ListTablesResponse::tableNames);

        // When future is complete (either successfully or in error) handle the
 response
        tableNames.whenComplete((tables, err) -> {
            try {
                if (tables != null) {
                    tables.forEach(System.out::println);
                } else {
                    // Handle error
                    err.printStackTrace();
                }
```

```
            } finally {
                // Lets the application shut down. Only close the client when you are
 completely done with it.
                client.close();
            }
        });
        tableNames.join();
    }
}
```

The following code example shows you how to retrieve an Item from a table by using the Asynchronous client. Invoke the `getItem` method of the DynamoDbAsyncClient and pass it a [GetItemRequest](#) object with the table name and primary key value of the item you want. This is typically how you pass data that the operation requires. In this example, notice that a String value is passed.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
    public static void getItem(DynamoDbAsyncClient client, String tableName, String
 key,  String keyVal) {

        HashMap<String, AttributeValue> keyToGet =
                new HashMap<String, AttributeValue>();

        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal).build());

        try {

            // Create a GetItemRequest instance
```

```
            GetItemRequest request = GetItemRequest.builder()
                    .key(keyToGet)
                    .tableName(tableName)
                    .build();

            // Invoke the DynamoDbAsyncClient object's getItem
            java.util.Collection<AttributeValue> returnedItem =
 client.getItem(request).join().item().values();

            // Convert Set to Map
            Map<String, AttributeValue> map =
 returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
            Set<String> keys = map.keySet();
            for (String sinKey : keys) {
                System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

See the complete example on GitHub.

Streaming operations

For streaming operations, you must provide an AsyncRequestBody to provide the content incrementally, or an AsyncResponseTransformer to receive and process the response.

The following example uploads a file to Amazon S3 asynchronously by using the `PutObject` operation.

Imports

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

Code

```
/**
 * To run this AWS code example, ensure that you have setup your development
 environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class S3AsyncOps {

     public static void main(String[] args) {

         final String USAGE = "\n" +
                 "Usage:\n" +
                 "    S3AsyncOps <bucketName> <key> <path>\n\n" +
                 "Where:\n" +
                 "    bucketName - the name of the Amazon S3 bucket (for example,
 bucket1). \n\n" +
                 "    key - the name of the object (for example, book.pdf). \n" +
                 "    path - the local path to the file (for example, C:/AWS/book.pdf).
 \n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
             System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
                .region(region)
                .build();

        PutObjectRequest objectRequest = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        // Put the object into the bucket
```

```
        CompletableFuture<PutObjectResponse> future = client.putObject(objectRequest,
                AsyncRequestBody.fromFile(Paths.get(path))
        );
        future.whenComplete((resp, err) -> {
            try {
                if (resp != null) {
                    System.out.println("Object uploaded. Details: " + resp);
                } else {
                    // Handle error
                    err.printStackTrace();
                }
            } finally {
                // Only close the client when you are completely done with it
                client.close();
            }
        });

        future.join();
    }
}
```

The following example gets a file from Amazon S3 asynchronously by using the `GetObject` operation.

Imports

```
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

Code

```
/**
 * To run this AWS code example, ensure that you have setup your development
 environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */

public class S3AsyncStreamOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    S3AsyncStreamOps <bucketName> <objectKey> <path>\n\n" +
                "Where:\n" +
                "    bucketName - the name of the Amazon S3 bucket (for example,
 bucket1). \n\n" +
                "    objectKey - the name of the object (for example, book.pdf). \n" +
                "    path - the local path to the file (for example, C:/AWS/book.pdf).
 \n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
         }

        String bucketName = args[0];
        String objectKey = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
                .region(region)
                .build();

        GetObjectRequest objectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

        CompletableFuture<GetObjectResponse> futureGet =
 client.getObject(objectRequest,
                AsyncResponseTransformer.toFile(Paths.get(path)));

        futureGet.whenComplete((resp, err) -> {
            try {
                if (resp != null) {
                    System.out.println("Object downloaded. Details: "+resp);
                } else {
```

```
                    err.printStackTrace();
                }
            } finally {
               // Only close the client when you are completely done with it
                client.close();
            }
        });
        futureGet.join();
    }
}
```

Advanced operations

The AWS SDK for Java 2.x uses Netty, an asynchronous event-driven network application framework, to handle I/O threads. The AWS SDK for Java 2.x creates an `ExecutorService` behind Netty, to complete the futures returned from the HTTP client request through to the Netty client. This abstraction reduces the risk of an application breaking the async process if developers choose to stop or sleep threads. By default, each asynchronous client creates a threadpool based on the number of processors and manages the tasks in a queue within the `ExecutorService`.

Advanced users can specify their thread pool size when creating an asynchronous client using the following option when building.

Code

```
S3AsyncClient clientThread = S3AsyncClient.builder()
  .asyncConfiguration(
    b -> b.advancedOption(SdkAdvancedAsyncClientOption
      .FUTURE_COMPLETION_EXECUTOR,
      Executors.newFixedThreadPool(10)
    )
  )
  .build();
```

To optimize performance, you can manage your own thread pool executor, and include it when configuring your client.

```
ThreadPoolExecutor executor = new ThreadPoolExecutor(50, 50,
    10, TimeUnit.SECONDS,
    new LinkedBlockingQueue<>(<custom_value>),
    new ThreadFactoryBuilder()
      .threadNamePrefix("sdk-async-response").build());
```

```
// Allow idle core threads to time out
executor.allowCoreThreadTimeOut(true);

S3AsyncClient clientThread = S3AsyncClient.builder()
  .asyncConfiguration(
    b -> b.advancedOption(SdkAdvancedAsyncClientOption
      .FUTURE_COMPLETION_EXECUTOR,
      executor
    )
  )
  .build();
```

Work with HTTP/2 in the AWS SDK for Java

HTTP/2 is a major revision of the HTTP protocol. This new version has several enhancements to improve performance:

- Binary data encoding provides more efficient data transfer.
- Header compression reduces the overhead bytes downloaded by the client, helping get the content to the client sooner. This is especially useful for mobile clients that are already constrained on bandwidth.
- Bidirectional asynchronous communication (multiplexing) allows multiple requests and response messages between the client and AWS to be in flight at the same time over a single connection, instead of over multiple connections, which improves performance.

Developers upgrading to the latest SDKs will automatically use HTTP/2 when it's supported by the service they're working with. New programming interfaces seamlessly take advantage of HTTP/2 features and provide new ways to build applications.

The AWS SDK for Java 2.x features new APIs for event streaming that implement the HTTP/2 protocol. For examples of how to use these new APIs, see Working with Kinesis.

Use SDK metrics from the AWS SDK for Java

With the AWS SDK for Java 2.x, you can collect metrics about the service clients in your application, analyze the output in Amazon CloudWatch, and then act on it.

By default, metrics collection is disabled in the SDK. This topic helps you to enable and configure it.

Prerequisites

Before you can enable and use metrics, you must complete the following steps:

- Complete the steps in *Setup*.
- Configure your project dependencies (for example, in your `pom.xml` or `build.gradle` file) to use version `2.14.0` or later of the AWS SDK for Java.

  To enabling publishing of metrics to CloudWatch, also include the artifactId `cloudwatch-metric-publisher` with the version number `2.14.0` or later in your project's dependencies.

  For example:

```
<project>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.14.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>cloudwatch-metric-publisher</artifactId>
      <version>2.14.0</version>
   </dependency>
  </dependencies>
</project>
```

- Enable `cloudwatch:PutMetricData` permissions for the IAM identity used by the metrics publisher to allow the SDK for Java to write metrics.

How to enable metrics collection

You can enable metrics in your application for a service client or on individual requests.

Enable metrics for a specific request

The following class shows how to enable the CloudWatch metrics publisher for a request to Amazon DynamoDB. It uses the default metrics publisher configuration.

```
import software.amazon.awssdk.metrics.MetricPublisher;
import software.amazon.awssdk.metrics.publishers.cloudwatch.CloudWatchMetricPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;

public class DefaultConfigForRequest {
    // Use one MetricPublisher for your application. It can be used with requests or
 service clients.
    static MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

    public static void main(String[] args) {
        DynamoDbClient ddb = DynamoDbClient.create();
        // Publish metrics the for ListTables operation.
        ddb.listTables(ListTablesRequest.builder()
            .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
            .build());

        // Perform more work in your application.

        // A MetricsPublisher has its own lifecycle independent of any service client
 or request that uses it.
        // If you no longer need the publisher, close it to free up resources.
        metricsPub.close();  // All metrics stored in memory are flushed to CloudWatch.

        // Perform more work with the DynamoDbClient instance without publishing
 metrics.
        // Close the service client when you no longer need it.
        ddb.close();
    }
}
```

Important

Make sure your application calls `close` on the [`MetricPublisher`] instance when the service client is no longer in use. Failure to do so results in possible thread or file descriptor leaks.

Enable summary metrics for a specific service client

The following code snippet shows how to enable a CloudWatch metrics publisher with default settings for a service client.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

DynamoDbClient ddb = DynamoDbClient.builder()
          .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
          .build();
```

Customize metrics publisher

The following class demonstrates how to set up a custom configuration for the metrics publisher for a specific service client. The customizations include loading a specific profile, specifying a AWS Region where the metrics publisher sends requests, and customizing how often the publisher sends metrics to CloudWatch.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.metrics.CoreMetric;
import software.amazon.awssdk.metrics.MetricPublisher;
import software.amazon.awssdk.metrics.publishers.cloudwatch.CloudWatchMetricPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import java.time.Duration;

public class CustomConfigForDDBClient {
    // Use one MetricPublisher for your application. It can be used with requests or
 service clients.
    static MetricPublisher metricsPub = CloudWatchMetricPublisher.builder()
        .cloudWatchClient(CloudWatchAsyncClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create("cloudwatch"))
            .build())
        .uploadFrequency(Duration.ofMinutes(5))
        .maximumCallsPerUpload(100)
        .namespace("ExampleSDKV2Metrics")
        .detailedMetrics(CoreMetric.API_CALL_DURATION)
        .build();
```

```
    public static void main(String[] args) {
        DynamoDbClient ddb = DynamoDbClient.builder()
            .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
            .build();
        // Publish metrics for DynamoDB operations.
        ddb.listTables();
        ddb.describeEndpoints();
        ddb.describeLimits();
        // Perform more work in your application.

        // A MetricsPublisher has its own lifecycle independent of any service client
 or request that uses it.
        // If you no longer need the publisher, close it to free up resources.
        metricsPub.close();  // All metrics stored in memory are flushed to CloudWatch.


        // Perform more work with the DynamoDbClient instance without publishing
 metrics.
        // Close the service client when you no longer need it.
        ddb.close();
    }
}
```

The customizations shown in the previous snippet have the following effects.

- The `cloudWatchClient` method lets you customize the CloudWatch client used to send metrics. In this example, we use a different region from the default of *us-east-1* where the client sends metrics. We also use a different named profile, *cloudwatch*, whose credentials will be used to authenticate requests to CloudWatch. Those credentials must have permissions to `cloudwatch:PutMetricData`.
- The `uploadFrequency` method allows you to specify how frequently the metrics publisher uploads metrics to CloudWatch. The default is once a minute.
- The `maximumCallsPerUpload` method limits the number of calls made per upload. The default is unlimited.
- By default, the SDK for Java 2.x publishes metrics under the namespace `AwsSdk/JavaSdk2`. You can use the `namespace` method to specify a different value.
- By default, the SDK publishes summary metrics. Summary metrics consist of average, minimum, maximum, sum, and sample count. By specifying one or more SDK metrics in the `detailedMetrics` method, the SDK publishes additional data for each metric. This additional data enables percentile statistics like p90 and p99 that you can query in CloudWatch. The

detailed metrics are especially useful for latency metrics like `APICallDuration`, which measures the end-to-end latency for SDK client requests. You can use fields of the `CoreMetric` class to specify other common SDK metrics.

When are metrics available?

Metrics are generally available within 5-10 minutes after the SDK for Java emits them. For accurate and up-to-date metrics, check Cloudwatch at least 10 minutes after emitting the metrics from your Java applications.

What information is collected?

Metrics collection includes the following:

- Number of API requests, including whether they succeed or fail
- Information about the AWS services you call in your API requests, including exceptions returned
- The duration for various operations such as Marshalling, Signing, and HTTP requests
- HTTP client metrics, such as the number of open connections, the number of pending requests, and the name of the HTTP client used

> **Note**
>
> The metrics available vary by HTTP client.

For a complete list, see Service client metrics.

How can I use this information?

You can use the metrics the SDK collects to monitor the service clients in your application. You can look at overall usage trends, identify anomalies, review service client exceptions returned, or to dig in to understand a particular issue. Using Amazon CloudWatch, you can also create alarms to notify you as soon as your application reaches a condition that you define.

For more information, see Using Amazon CloudWatch Metrics and Using Amazon CloudWatch Alarms in the Amazon CloudWatch User Guide.

Service client metrics

With the AWS SDK for Java 2.x, you can collect metrics from the service clients in your application and then publish (output) those metrics to Amazon CloudWatch.

These tables list the metrics that you can collect and any HTTP client usage requirement.

For more information about enabling and configuring metrics for the SDK, see Enabling SDK metrics.

Metrics collected with each request

Metric name	Description	Type
ApiCallDuration	The total time taken to finish a request (inclusive of all retries).	Duration
ApiCallSuccessful	True if the API call was successful; false if not.	Boolean
CredentialsFetchDuration	The time taken to fetch AWS signing credentials for the request.	Duration
EndpointResolveDuration	The duration of time it took to resolve the endpoint used for the API call.	Duration
MarshallingDuration	The time it takes to marshall an SDK request to an HTTP request.	Duration
OperationName	The name of the AWS API the request is made to.	String
RetryCount	Number of times the SDK retried the API call.	Integer

Metric name	Description	Type
ServiceId	Service ID of the AWS service that the API request is made against.	String
TokenFetchDuration	The time taken to fetch token signing credentials for the request.	Duration

Metrics collected for each request attempt

Each API call might require multiple attempts before a response is received. These metrics are collected for each attempt.

Core metrics

Metric name	Description	Type
AwsExtendedRequestId	The extended request ID of the service request.	String
AwsRequestId	The request ID of the service request.	String
BackoffDelayDuration	The duration of time the SDK waited before this API call attempt.	Duration
ErrorType	The type of error that occurred for a call attempt.	String
ReadThroughput	The read throughput of the client.	Double
ServiceCallDuration	The time it takes to connect to the service, send the request, and receive the HTTP	Duration

Metric name	Description	Type
	status code and header from the response.	
SigningDuration	The time it takes to sign the HTTP request.	Duration
TimeToFirstByte	Elapsed time from sending the HTTP request (includin g acquiring a connection) to receiving the first byte of the headers in the response.	Duration
TimeToLastByte	Elapsed time from sending the HTTP request (includin g acquiring a connection) to receiving the last byte of the response.	Duration
UnmarshallingDuration	The time it takes to unmarshall an HTTP response to an SDK response.	Duration

HTTP Metrics

Metric name	Description	Type	HTTP client required*
AvailableConcurrency	The number of remaining concurren t requests that can be supported by the HTTP client without needing to establish another connection.	Integer	Apache, Netty, CRT

Metric name	Description	Type	HTTP client required*
ConcurrencyAcquireDuration	The time taken to acquire a channel from the connection pool.	Duration	Apache, Netty, CRT
HttpClientName	The name of the HTTP being use for the request.	String	Apache, Netty, CRT
HttpStatusCode	The status code returned with the HTTP response.	Integer	Any
LeasedConcurrency	The number of request currently being executed by the HTTP client.	Integer	Apache, Netty, CRT
LocalStreamWindowSize	The local HTTP/2 window size in bytes for the stream that this request was executed on.	Integer	Netty
MaxConcurrency	The max number of concurrent requests supported by the HTTP client.	Integer	Apache, Netty, CRT

Metric name	Description	Type	HTTP client required*
PendingConcurrencyAcquires	The number of requests that are blocked, waiting for another TCP connection or a new stream to be available from the connection pool.	Integer	Apache, Netty, CRT
RemoteStreamWindowSize	The remote HTTP/2 window size in bytes for the stream that this request was executed on.	Integer	Netty

The terms used in the column mean:

- Apache: the Apache-based HTTP client (`ApacheHttpClient`)
- Netty: the Netty-based HTTP client (`NettyNioAsyncHttpClient`)
- CRT: the AWS CRT-based HTTP client (`AwsCrtAsyncHttpClient`)
- Any: the collection of metric data does not depend on the HTTP client; this includes the URLConnection-based HTTP client (`UrlConnectionHttpClient`)

Work with AWS services using the AWS SDK for Java 2.x

This section provides short tutorials and guidance for how to work with select AWS services. For a complete set of examples, see the Code Examples section.

Topics

Work with CloudWatch

This section provides examples of programming Amazon CloudWatch by using the AWS SDK for Java 2.x.

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications. CloudWatch alarms send notifications or automatically make changes to the resources you are monitoring based on rules that you define.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- Get metrics from CloudWatch
- Publish custom metric data to CloudWatch
- Work with CloudWatch alarms
- Use Amazon CloudWatch Events

Get metrics from CloudWatch

Listing metrics

To list CloudWatch metrics, create a ListMetricsRequest and call the CloudWatchClient's `listMetrics` method. You can use the `ListMetricsRequest` to filter the returned metrics by namespace, metric name, or dimensions.

> **Note**
>
> A list of metrics and dimensions that are posted by AWS services can be found within the Amazon CloudWatch Metrics and Dimensions Reference in the Amazon CloudWatch User Guide.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
```

Code

```
    public static void listMets( CloudWatchClient cw, String namespace) {

        boolean done = false;
        String nextToken = null;

        try {
            while(!done) {
```

```
                ListMetricsResponse response;

                if (nextToken == null) {
                   ListMetricsRequest request = ListMetricsRequest.builder()
                        .namespace(namespace)
                        .build();

                 response = cw.listMetrics(request);
                } else {
                  ListMetricsRequest request = ListMetricsRequest.builder()
                        .namespace(namespace)
                        .nextToken(nextToken)
                        .build();

                response = cw.listMetrics(request);
            }

            for (Metric metric : response.metrics()) {
                System.out.printf(
                        "Retrieved metric %s", metric.metricName());
                System.out.println();
            }

            if(response.nextToken() == null) {
                done = true;
            } else {
                nextToken = response.nextToken();
            }
        }

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

The metrics are returned in a ListMetricsResponse by calling its `getMetrics` method.

The results may be *paged*. To retrieve the next batch of results, call `nextToken` on the response object and use the token value to build a new request object. Then call the `listMetrics` method again with the new request.

See the complete example on GitHub.

More information

- ListMetrics in the Amazon CloudWatch API Reference

Publish custom metric data to CloudWatch

A number of AWS services publish their own metrics in namespaces beginning with " AWS " You can also publish custom metric data using your own namespace (as long as it doesn't begin with " AWS ").

Publish custom metric data

To publish your own metric data, call the CloudWatchClient's `putMetricData` method with a PutMetricDataRequest. The `PutMetricDataRequest` must include the custom namespace to use for the data, and information about the data point itself in a MetricDatum object.

> **Note**
>
> You cannot specify a namespace that begins with " AWS ". Namespaces that begin with " AWS " are reserved for use by Amazon Web Services products.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
```

Code

```
    public static void putMetData(CloudWatchClient cw, Double dataPoint ) {
```

```
        try {
            Dimension dimension = Dimension.builder()
                    .name("UNIQUE_PAGES")
                    .value("URLS")
                    .build();

            // Set an Instant object
            String time =
 ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                .metricName("PAGES_VISITED")
                .unit(StandardUnit.NONE)
                .value(dataPoint)
                .timestamp(instant)
                .dimensions(dimension).build();

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                .namespace("SITE/TRAFFIC")
                .metricData(datum).build();

            cw.putMetricData(request);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.printf("Successfully put data point %f", dataPoint);
    }
```

See the complete example on GitHub.

More information

- Use Amazon CloudWatch Metrics in the Amazon CloudWatch User Guide.
- AWS Namespaces in the Amazon CloudWatch User Guide.
- PutMetricData in the Amazon CloudWatch API Reference.

Work with CloudWatch alarms

Create an alarm

To create an alarm based on a CloudWatch metric, call the CloudWatchClient's `putMetricAlarm` method with a [PutMetricAlarmRequest](#) filled with the alarm conditions.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
```

Code

```
    public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
 instanceId) {

        try {
            Dimension dimension = Dimension.builder()
                .name("InstanceId")
                .value(instanceId).build();

            PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
                .alarmName(alarmName)
                .comparisonOperator(
                        ComparisonOperator.GREATER_THAN_THRESHOLD)
                .evaluationPeriods(1)
                .metricName("CPUUtilization")
                .namespace("AWS/EC2")
                .period(60)
                .statistic(Statistic.AVERAGE)
                .threshold(70.0)
                .actionsEnabled(false)
                .alarmDescription(
                        "Alarm when server CPU utilization exceeds 70%")
                .unit(StandardUnit.SECONDS)
                .dimensions(dimension)
```

```
                .build();

            cw.putMetricAlarm(request);
            System.out.printf(
                    "Successfully created alarm with name %s", alarmName);

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

See the complete example on GitHub.

List alarms

To list the CloudWatch alarms that you have created, call the CloudWatchClient's `describeAlarms` method with a DescribeAlarmsRequest that you can use to set options for the result.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
```

Code

```
    public static void desCWAlarms( CloudWatchClient cw) {

        try {

            boolean done = false;
            String newToken = null;

            while(!done) {
                DescribeAlarmsResponse response;

                if (newToken == null) {
```

```
                    DescribeAlarmsRequest request =
DescribeAlarmsRequest.builder().build();
                    response = cw.describeAlarms(request);
                } else {
                    DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
                        .nextToken(newToken)
                        .build();
                    response = cw.describeAlarms(request);
                }

                for(MetricAlarm alarm : response.metricAlarms()) {
                    System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
                }

                if(response.nextToken() == null) {
                    done = true;
                } else {
                    newToken = response.nextToken();
                }
            }

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

The list of alarms can be obtained by calling `MetricAlarms` on the [DescribeAlarmsResponse](#) that is returned by `describeAlarms`.

The results may be *paged*. To retrieve the next batch of results, call `nextToken` on the response object and use the token value to build a new request object. Then call the `describeAlarms` method again with the new request.

> **Note**
>
> You can also retrieve alarms for a specific metric by using the CloudWatchClient's `describeAlarmsForMetric` method. Its use is similar to `describeAlarms`.

See the [complete example](#) on GitHub.

Delete alarms

To delete CloudWatch alarms, call the CloudWatchClient's `deleteAlarms` method with a DeleteAlarmsRequest containing one or more names of alarms that you want to delete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
```

Code

```
    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {

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

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

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

See the complete example on GitHub.

More information

- Using Amazon CloudWatch alarms in the Amazon CloudWatch User Guide
- PutMetricAlarm in the Amazon CloudWatch API Reference
- DescribeAlarms in the Amazon CloudWatch API Reference
- DeleteAlarms in the Amazon CloudWatch API Reference

Use Amazon CloudWatch Events

CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources to Amazon EC2 instances, Lambda functions, Kinesis streams, Amazon ECS tasks, Step Functions state machines, Amazon SNS topics, Amazon SQS queues, or built-in targets. You can match events and route them to one or more target functions or streams by using simple rules.

Amazon EventBridge is the evolution of CloudWatch Events. Both services use the same API, so you can continue using the CloudWatch Events client provided by the SDK or migrate to the SDK for Java's EventBridge client for CloudWatch Events functionality. CloudWatch Events User Guide documentation and API reference are now available through the EventBridge documentation sites.

Add events

To add custom CloudWatch events, call the `CloudWatchEventsClient's` `putEvents` method with a `PutEventsRequest` object that contains one or more `PutEventsRequestEntry` objects that provide details about each event. You can specify several parameters for the entry such as the source and type of the event, resources associated with the event, and so on.

Note

You can specify a maximum of 10 events per call to `putEvents`.

Imports

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;
```

Code

```
    public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn ) {

        try {

            final String EVENT_DETAILS =
                "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
```

```
                    .detail(EVENT_DETAILS)
                    .detailType("sampleSubmitted")
                    .resources(resourceArn)
                    .source("aws-sdk-java-cloudwatch-example")
                    .build();

            PutEventsRequest request = PutEventsRequest.builder()
                    .entries(requestEntry)
                    .build();

            cwe.putEvents(request);
            System.out.println("Successfully put CloudWatch event");

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

See the complete example on GitHub.

Add rules

To create or update a rule, call the `CloudWatchEventsClient’s` `putRule` method with a `PutRuleRequest` with the name of the rule and optional parameters such as the event pattern, IAM role to associate with the rule, and a scheduling expression that describes how often the rule is run.

Imports

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;
```

Code

```
    public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
 roleArn) {

        try {
            PutRuleRequest request = PutRuleRequest.builder()
```

```
                .name(ruleName)
                .roleArn(roleArn)
                .scheduleExpression("rate(5 minutes)")
                .state(RuleState.ENABLED)
                .build();

            PutRuleResponse response = cwe.putRule(request);
            System.out.printf(
                    "Successfully created CloudWatch events rule %s with arn %s",
                    roleArn, response.ruleArn());
        } catch (
            CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Add targets

Targets are the resources that are invoked when a rule is triggered. Example targets include Amazon EC2 instances, Lambda functions, Kinesis streams, Amazon ECS tasks, Step Functions state machines, and built-in targets.

To add a target to a rule, call the `CloudWatchEventsClient’s` `putTargets` method with a `PutTargetsRequest` containing the rule to update and a list of targets to add to the rule.

Imports

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;
```

Code

```
    public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName, String
 functionArn, String targetId ) {

        try {
            Target target = Target.builder()
```

```
                .arn(functionArn)
                .id(targetId)
                .build();

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

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

See the complete example on GitHub.

More information

- Adding Events with PutEvents in the Amazon EventBridge User Guide
- Schedule Expressions for Rules in the Amazon EventBridge User Guide
- Event Types for CloudWatch Events in the Amazon EventBridge User Guide
- Event Patterns in the Amazon EventBridge User Guide
- PutEvents in the Amazon EventBridge API Reference
- PutTargets in the Amazon EventBridge API Reference
- PutRule in the Amazon EventBridge API Reference

AWS database services and AWS SDK for Java 2.x

AWS offers several database types: relational, key-value, in-memory, document, and several others. The SDK for Java 2.x support varies depending the nature of the database service in AWS.

Some database services, for example Amazon DynamoDB service, have web service APIs to manage the AWS resource (database) as well as web service APIs to interact with the data. In the SDK for Java 2.x these types of services have dedicated service clients, for example DynamoDBClient.

Other database services have web service APIs that interact with the resource, such the Amazon DocumentDB API (for cluster, instance and resource management), but do not have a web service API for working with the data. The SDK for Java 2.x has a corresponding DocDbClient interface for working with the resource. However, you need another Java API, such as MongoDB for Java to work with the data.

Use the examples below to learn how you use the SDK for Java 2.x service clients with the different types of databases.

Amazon DynamoDB examples

Working with the data	Working with the database
SDK service client: DynamoDbClient Example: React/Spring REST application using DynamoDB Examples: Several DynamoDB examples SDK service client: DynamoDbEnhancedClient Example: React/Spring REST application using DynamoDB Examples: Several DynamoDB examples (names starting with 'Enhanced")	SDK service client: DynamoDbClient Examples: CreateTable, ListTables, DeleteTable

See additional DynamoDB examples in the guided code examples section of this guide.

Amazon RDS examples

Working with the data	Working with the database
Non-SDK API: JDBC, database-specific SQL flavor; your code manages database connectio ns or a connection pool.	SDK service client: RdsClient

Working with the data	Working with the database
Example: React/Spring REST application using MySQL	Examples: Several RdsClient examples

Amazon Redshift examples

Working with the data	Working with the database
SDK service client: RedshiftDataClient	SDK service client: RedshiftClient
Examples: Several RedshiftDataClient examples	Examples: Several RedshiftClient examples
Example: React/Spring REST application using RedshiftDataClient	

Amazon Aurora Serverless v2 examples

Working with the data	Working with the database
SDK service client: RdsDataClient	SDK service client: RdsClient
Example: React/Spring REST application using RdsDataClient	Examples: Several RdsClient examples

Amazon DocumentDB examples

Working with the data	Working with the database
Non-SDK API: MongoDB-specific Java library (for example MongoDB for Java); your code manages database connections or a connectio n pool.	SDK service client: DocDbClient

Working with the data	Working with the database
Examples: DocumentDB (Mongo) Developer Guide (select 'Java' tab)	

Work with DynamoDB

This section provides examples that show you how to work with DynamoDB.

The following examples use the standard, low-level DynamoDB client (`DynamoDbClient`) of the AWS SDK for Java 2.x.

- the section called “Work with tables in DynamoDB”
- the section called “Work with items in DynamoDB”

The SDK also offers the DynamoDB Enhanced Client that provides a high-level, object-oriented approach for working with DynamoDB. The following section discusses this client in depth.

- the section called “ Map objects to DynamoDB items”

Work with tables in DynamoDB

Tables are the containers for all items in a DynamoDB database. Before you can add or remove data from DynamoDB, you must create a table.

For each table, you must define:

- A table *name* that is unique for your account and Region.
- A *primary key* for which every value must be unique; no two items in your table can have the same primary key value.

  A primary key can be *simple*, consisting of a single partition (HASH) key, or *composite*, consisting of a partition and a sort (RANGE) key.

  Each key value has an associated *data type*, enumerated by the `ScalarAttributeType` class. The key value can be binary (B), numeric (N), or a string (S). For more information, see Naming Rules and Data Types in the Amazon DynamoDB Developer Guide.

- *Provisioned throughput* are values that define the number of reserved read/write capacity units for the table.

> **Note**
>
> Amazon DynamoDB pricing is based on the provisioned throughput values that you set on your tables, so reserve only as much capacity as you think you'll need for your table.

Provisioned throughput for a table can be modified at any time, so you can adjust capacity as your needs change.

Create a table

Use the `DynamoDbClient’s` `createTable` method to create a new DynamoDB table. You need to construct table attributes and a table schema, both of which are used to identify the primary key of your table. You must also supply initial provisioned throughput values and a table name.

> **Note**
>
> If a table with the name you chose already exists, an `DynamoDbException` is thrown.

Create a table with a simple primary key

This code creates a table with one attribute that is the table's simple primary key. the example uses `AttributeDefinition` and `KeySchemaElement` objects for the `CreateTableRequest`.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
```

```
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

Code

```
    public static String createTable(DynamoDbClient ddb, String tableName, String key)
 {

        DynamoDbWaiter dbWaiter = ddb.waiter();
        CreateTableRequest request = CreateTableRequest.builder()
                .attributeDefinitions(AttributeDefinition.builder()
                        .attributeName(key)
                        .attributeType(ScalarAttributeType.S)
                        .build())
                .keySchema(KeySchemaElement.builder()
                        .attributeName(key)
                        .keyType(KeyType.HASH)
                        .build())
                .provisionedThroughput(ProvisionedThroughput.builder()
                        .readCapacityUnits(new Long(10))
                        .writeCapacityUnits(new Long(10))
                        .build())
                .tableName(tableName)
                .build();

        String newTable ="";
        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            // Wait until the Amazon DynamoDB table is created
            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);

            newTable = response.tableDescription().tableName();
            return newTable;
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
       return "";
    }
```

See the complete example on GitHub.

Create a table with a composite primary key

The following example creates a table with two attributes. Both attributes are used for the composite primary key.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```

Code

```
    public static String createTableComKey(DynamoDbClient ddb, String tableName) {
        CreateTableRequest request = CreateTableRequest.builder()
                .attributeDefinitions(
                        AttributeDefinition.builder()
                                .attributeName("Language")
                                .attributeType(ScalarAttributeType.S)
                                .build(),
                        AttributeDefinition.builder()
                                .attributeName("Greeting")
                                .attributeType(ScalarAttributeType.S)
                                .build())
                .keySchema(
                        KeySchemaElement.builder()
                                .attributeName("Language")
```

```
                                .keyType(KeyType.HASH)
                                .build(),
                        KeySchemaElement.builder()
                                .attributeName("Greeting")
                                .keyType(KeyType.RANGE)
                                .build())
                .provisionedThroughput(
                        ProvisionedThroughput.builder()
                                .readCapacityUnits(new Long(10))
                                .writeCapacityUnits(new Long(10)).build())
                .tableName(tableName)
                .build();

        String tableId = "";

        try {
             CreateTableResponse result = ddb.createTable(request);
             tableId = result.tableDescription().tableId();
             return tableId;
         } catch (DynamoDbException e) {
             System.err.println(e.getMessage());
             System.exit(1);
         }
        return "";
    }
```

See the [complete example](#) on GitHub.

List tables

You can list the tables in a particular Region by calling the `DynamoDbClient’s` `listTables` method.

> **Note**
>
> If the named table doesn’t exist for your account and Region, a `ResourceNotFoundException` is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.List;
```

Code

```
    public static void listAllTables(DynamoDbClient ddb){

        boolean moreTables = true;
        String lastName = null;

        while(moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();

                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }
            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
                System.exit(1);
```

```
            }
        }
        System.out.println("\nDone!");
    }
```

By default, up to 100 tables are returned per call—use `lastEvaluatedTableName` on the returned [`ListTablesResponse`]() object to get the last table that was evaluated. You can use this value to start the listing after the last returned value of the previous listing.

See the [complete example]() on GitHub.

Describe (get information about) a table

Use the `DynamoDbClient’s` `describeTable` method to get information about a table.

> **Note**
>
> If the named table doesn’t exist for your account and Region, a [`ResourceNotFoundException`]() is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;
```

Code

```
    public static void describeDymamoDBTable(DynamoDbClient ddb,String tableName ) {

        DescribeTableRequest request = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();
```

```
        try {
            TableDescription tableInfo =
                    ddb.describeTable(request).table();

            if (tableInfo != null) {
                System.out.format("Table name  : %s\n",
                        tableInfo.tableName());
                System.out.format("Table ARN   : %s\n",
                        tableInfo.tableArn());
                System.out.format("Status      : %s\n",
                        tableInfo.tableStatus());
                System.out.format("Item count  : %d\n",
                        tableInfo.itemCount().longValue());
                System.out.format("Size (bytes): %d\n",
                        tableInfo.tableSizeBytes().longValue());

                ProvisionedThroughputDescription throughputInfo =
                        tableInfo.provisionedThroughput();
                System.out.println("Throughput");
                System.out.format("  Read Capacity : %d\n",
                        throughputInfo.readCapacityUnits().longValue());
                System.out.format("  Write Capacity: %d\n",
                        throughputInfo.writeCapacityUnits().longValue());

                List<AttributeDefinition> attributes =
                        tableInfo.attributeDefinitions();
                System.out.println("Attributes");

                for (AttributeDefinition a : attributes) {
                    System.out.format("  %s (%s)\n",
                            a.attributeName(), a.attributeType());
                }
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("\nDone!");
    }
```

See the complete example on GitHub.

Modify (update) a table

You can modify your table's provisioned throughput values at any time by calling the `DynamoDbClient`'s `updateTable` method.

> **Note**
>
> If the named table doesn't exist for your account and Region, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
    public static void updateDynamoDBTable(DynamoDbClient ddb,
                                           String tableName,
                                           Long readCapacity,
                                           Long writeCapacity) {

        System.out.format(
                "Updating %s with new provisioned throughput values\n",
                tableName);
        System.out.format("Read capacity : %d\n", readCapacity);
        System.out.format("Write capacity : %d\n", writeCapacity);

        ProvisionedThroughput tableThroughput = ProvisionedThroughput.builder()
                .readCapacityUnits(readCapacity)
                .writeCapacityUnits(writeCapacity)
                .build();

        UpdateTableRequest request = UpdateTableRequest.builder()
                .provisionedThroughput(tableThroughput)
                .tableName(tableName)
                .build();
```

```
        try {
            ddb.updateTable(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }
```

See the complete example on GitHub.

Delete a table

To delete a table, call `DynamoDbClient’s deleteTable` method and provide the table’s name.

> **Note**
>
> If the named table doesn’t exist for your account and Region, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
```

Code

```
    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

        DeleteTableRequest request = DeleteTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            ddb.deleteTable(request);
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName +" was successfully deleted!");
    }
```

See the complete example on GitHub.

More information

- Guidelines for Working with Tables in the Amazon DynamoDB Developer Guide
- Working with Tables in DynamoDB in the Amazon DynamoDB Developer Guide

Work with items in DynamoDB

In DynamoDB, an item is a collection of *attributes*, each of which has a *name* and a *value*. An attribute value can be a scalar, set, or document type. For more information, see Naming Rules and Data Types in the Amazon DynamoDB Developer Guide.

Retrieve (get) an item from a table

Call the DynamoDbClient's `getItem` method and pass it a GetItemRequest object with the table name and primary key value of the item you want. It returns a GetItemResponse object with all of the attributes for that item. You can specify one or more projection expressions in the `GetItemRequest` to retrieve specific attributes.

You can use the returned `GetItemResponse` object's `item()` method to retrieve a Map of key (String) and value (AttributeValue) pairs that are associated with the item.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
```

```
import java.util.Set;
```

Code

```
    public static void getDynamoDBItem(DynamoDbClient ddb,String tableName,String
 key,String keyVal ) {

        HashMap<String,AttributeValue> keyToGet = new HashMap<String,AttributeValue>();

        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal).build());

        GetItemRequest request = GetItemRequest.builder()
                .key(keyToGet)
                .tableName(tableName)
                .build();

        try {
            Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", key);
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Retrieve (get) an item from a table using the asynchronous client

Invoke the `getItem` method of the DynamoDbAsyncClient and pass it a GetItemRequest object with the table name and primary key value of the item you want.

You can return a [Collection](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html) instance with all of the attributes for that item (refer to the following example).

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
    public static void getItem(DynamoDbAsyncClient client, String tableName, String
 key,  String keyVal) {

        HashMap<String, AttributeValue> keyToGet =
                new HashMap<String, AttributeValue>();

        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal).build());

        try {

            // Create a GetItemRequest instance
            GetItemRequest request = GetItemRequest.builder()
                    .key(keyToGet)
                    .tableName(tableName)
                    .build();

            // Invoke the DynamoDbAsyncClient object's getItem
            java.util.Collection<AttributeValue> returnedItem =
 client.getItem(request).join().item().values();

            // Convert Set to Map
            Map<String, AttributeValue> map =
 returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
            Set<String> keys = map.keySet();
            for (String sinKey : keys) {
```

```
                System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

See the complete example on GitHub.

Add a new item to a table

Create a Map of key-value pairs that represent the item's attributes. These must include values for the table's primary key fields. If the item identified by the primary key already exists, its fields are *updated* by the request.

> **Note**
>
> If the named table doesn't exist for your account and region, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;
```

Code

```
    public static void putItemInTable(DynamoDbClient ddb,
                                      String tableName,
                                      String key,
                                      String keyVal,
                                      String albumTitle,
                                      String albumTitleValue,
```

```
                                          String awards,
                                          String awardVal,
                                          String songTitle,
                                          String songTitleVal){

        HashMap<String,AttributeValue> itemValues = new
 HashMap<String,AttributeValue>();

        // Add all content to the table
        itemValues.put(key, AttributeValue.builder().s(keyVal).build());
        itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
        itemValues.put(albumTitle,
 AttributeValue.builder().s(albumTitleValue).build());
        itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

        PutItemRequest request = PutItemRequest.builder()
                .tableName(tableName)
                .item(itemValues)
                .build();

        try {
            ddb.putItem(request);
            System.out.println(tableName +" was successfully updated");

        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be found.
\n", tableName);
            System.err.println("Be sure that it exists and that you've typed its name
 correctly!");
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Update an existing item in a table

You can update an attribute for an item that already exists in a table by using the DynamoDbClient's `updateItem` method, providing a table name, primary key value, and a map of fields to update.

Note

If the named table doesn't exist for your account and region, or if the item identified by the primary key you passed in doesn't exist, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;
```

Code

```
    public static void updateTableItem(DynamoDbClient ddb,
                                       String tableName,
                                       String key,
                                       String keyVal,
                                       String name,
                                       String updateVal){

        HashMap<String,AttributeValue> itemKey = new HashMap<String,AttributeValue>();

        itemKey.put(key, AttributeValue.builder().s(keyVal).build());

        HashMap<String,AttributeValueUpdate> updatedValues =
                new HashMap<String,AttributeValueUpdate>();

        // Update the column specified by name with updatedVal
        updatedValues.put(name, AttributeValueUpdate.builder()
                .value(AttributeValue.builder().s(updateVal).build())
                .action(AttributeAction.PUT)
                .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
```

```
                .key(itemKey)
                .attributeUpdates(updatedValues)
                .build();

        try {
            ddb.updateItem(request);
        } catch (ResourceNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }
```

See the complete example on GitHub.

Delete an existing item in a table

You can delete an item that exists in a table by using the DynamoDbClient's `deleteItem` method and providing a table name as well as the primary key value.

> **Note**
>
> If the named table doesn't exist for your account and region, or if the item identified by the primary key you passed in doesn't exist, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;
```

Code

```
 public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {

       HashMap<String,AttributeValue> keyToGet =
               new HashMap<String,AttributeValue>();

       keyToGet.put(key, AttributeValue.builder()
               .s(keyVal)
               .build());

       DeleteItemRequest deleteReq = DeleteItemRequest.builder()
               .tableName(tableName)
               .key(keyToGet)
               .build();

       try {
           ddb.deleteItem(deleteReq);
       } catch (DynamoDbException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

See the complete example on GitHub.

More information

- Guidelines for Working with Items in the Amazon DynamoDB Developer Guide
- Working with Items in DynamoDB in the Amazon DynamoDB Developer Guide

Map Java objects to DynamoDB items with the AWS SDK for Java 2.x

The DynamoDB Enhanced Client API is a high-level library that is the successor to the `DynamoDBMapper` class of in the SDK for Java v1.x. It offers a straightforward way to map client-side classes to DynamoDB tables. You define the relationships between tables and their corresponding data classes in your code. After you define those relationships, you can intuitively perform various create, read, update, or delete (CRUD) operations on tables or items in DynamoDB.

The DynamoDB Enhanced Client API also includes the Enhanced Document API that enables you to work with document-type items that do not follow a defined schema.

The DynamoDB Enhanced Client API is discussed in the following topics.

- Get Started using the DynamoDB Enhanced Client API
- Learn the basics of the DynamoDB Enhanced Client API
- Use advanced mapping features
- Work with JSON documents with the Enhanced Document API for DynamoDB
- Use extensions
- Use the DynamoDB Enhanced Client API asynchronously
- Data class annotations

Get Started using the DynamoDB Enhanced Client API

The following tutorial introduces you to fundamentals that you need to work with the DynamoDB Enhanced Client API.

Add dependencies

To begin working with the DynamoDB Enhanced Client API in your project, add a dependency on the `dynamodb-enhanced` Maven artifact. This is shown in the following examples.

Maven

```
<project>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version><VERSION></version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb-enhanced</artifactId>
    </dependency>
```

```
  </dependencies>
  ...
</project>
```

Perform a search of the Maven central repository for the latest version and replace *<VERSION>* with this value.

Gradle

```
repositories {
    mavenCentral()
}
dependencies {
    implementation(platform("software.amazon.awssdk:bom:<VERSION>"))
    implementation("software.amazon.awssdk:dynamodb-enhanced")
    ...
}
```

Perform a search of the Maven central repository for the latest version and replace *<VERSION>* with this value.

Generate a `TableSchema` from a data class

A `TableSchema` enables the enhanced client to map DynamoDB attribute values to and from your client-side classes. In this tutorial, you learn about `TableSchemas` derived from a static data class and generated from code by using a builder.

Use an annotated data class

The SDK for Java 2.x includes a set of annotations that you can use with a data class to quickly generate a `TableSchema` for mapping your classes to tables.

Start by creating a data class that conforms to the JavaBean specification. The specification requires that a class has a no-argument public constructor and has getters and setters for each attribute in the class. Include a class-level annotation to indicate that the data class is a `DynamoDbBean`. Also, at a minimum, include a `DynamoDbPartitionKey` annotation on the getter or setter for the primary key attribute.

You can apply attribute-level annotations to getters or setters, but not both.

Note

The term `property` is normally used for a value encapsulated in a JavaBean. However, this guide uses the term `attribute` instead, to be consistent with the terminology used by DynamoDB.

The following `Customer` class shows annotations that link the class definition to a DynamoDB table.

`Customer` class

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.time.Instant;

@DynamoDbBean
public class Customer {

    private String id;
    private String name;
    private String email;
    private Instant regDate;

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

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

    public String getCustName() { return this.name; }

    public void setCustName(String name) { this.name = name; }

    @DynamoDbSortKey
    public String getEmail() { return this.email; }

    public void setEmail(String email) { this.email = email; }
```

```
    public Instant getRegistrationDate() { return this.regDate; }

    public void setRegistrationDate(Instant registrationDate) { this.regDate =
 registrationDate; }

    @Override
    public String toString() {
        return "Customer [id=" + id + ", name=" + name + ", email=" + email
                + ", regDate=" + regDate + "]";
    }
}
```

After you have created an annotated data class, use it to create the TableSchema, as shown in the following snippet.

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

A TableSchema is designed to be static and immutable. You can usually instantiate it at class-load time.

The static TableSchema.fromBean() factory method introspects the bean to generate the mapping of data class attributes (properties) to and from DynamoDB attributes.

For an example of working with a data model made up of several data classes, see the Person class in the ??? section.

Use a builder

You can skip the cost of bean introspection if you define the table schema in code. If you code the schema, your class does not need to follow JavaBean naming standards nor does it need to be annotated. The following example uses a builder and is equivalent to the Customer class example that uses annotations.

```
static final TableSchema<Customer> customerTableSchema =
                TableSchema.builder(Customer.class)
                        .newItemSupplier(Customer::new)
                        .addAttribute(String.class, a -> a.name("id")
                                .getter(Customer::getId)
                                .setter(Customer::setId)
```

```
                        .tags(StaticAttributeTags.primaryPartitionKey()))
                .addAttribute(String.class, a -> a.name("email")
                        .getter(Customer::getEmail)
                        .setter(Customer::setEmail)
                        .tags(StaticAttributeTags.primarySortKey()))
                .addAttribute(String.class, a -> a.name("name")
                        .getter(Customer::getCustName)
                        .setter(Customer::setCustName))
                .addAttribute(Instant.class, a -> a.name("registrationDate")
                        .getter(Customer::getRegistrationDate)
                        .setter(Customer::setRegistrationDate))
                .build();
```

Create an enhanced client and `DynamoDbTable`

Create an enhanced client

The DynamoDbEnhancedClient class or its asynchronous counterpart, DynamoDbEnhancedAsyncClient, is the entry point to working with the DynamoDB Enhanced Client API.

The enhanced client requires a standard `DynamoDbClient` to perform work. The API offers two ways to create a `DynamoDbEnhancedClient` instance. The first option, shown in the following snippet, creates a standard `DynamoDbClient` with default settings picked up from configuration settings.

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

If you want to configure the underlying standard client, you can supply it to the enhanced client's builder method as shown in the following snippet.

```
// Configure an instance of the standard DynamoDbClient.
DynamoDbClient standardClient = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

// Use the configured standard client with the enhanced client.
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
    .dynamoDbClient(standardClient)
    .build();
```

Create a `DynamoDbTable` instance

Think of a [`DynamoDbTable`] as the client-side representation of a DynamoDB table that uses the mapping functionality provided by a `TableSchema`. The `DynamoDbTable` class provides methods for CRUD operations that let you interact with a single DynamoDB table.

`DynamoDbTable<T>` is a generic class that takes a single type argument, whether it is a custom class or an `EnhancedDocument` when working with document-type items. This argument type establishes the relationship between the class that you use and the single DynamoDB table.

Use the `table()` factory method of the `DynamoDbEnhancedClient` to create a `DynamoDbTable` instance as shown in the following snippet.

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

`DynamoDbTable` instances are candidates for singletons because they are immutable and can be used throughout your application.

Your code now has an in-memory representation of a DynamoDB table that can work with `Customer` instances. The actual DynamoDB table might or might not exist. If the table named `Customer` already exists, you can begin performing CRUD operations against it. If it doesn't exist, use the `DynamoDbTable` instance to create the table as discussed in the next section.

Create a DynamoDB table if needed

After you have created a `DynamoDbTable` instance, use it to perform a *one-time* creation of a table in DynamoDB.

Create table example code

The following example creates a DynamoDB table based on the `Customer` data class.

This example creates a DynamoDB table with the name `Customer`—identical to the class name —but the table name can be something else. Whatever you name the table, you must use this name in additional applications to work with the table. Supply this name to the `table()` method anytime you create another `DynamoDbTable` object in order to work with the underlying DynamoDB table.

The Java lambda parameter, `builder`, passed to the `createTable` method lets you customize the table. In this example, provisioned throughput is configured. If you want to use default settings when you create a table, skip the builder as shown in the following snippet.

```
customerTable.createTable();
```

When default settings are used, values for provisioned throughput are not set. Instead, the billing mode for the table is set to on-demand.

The example also uses a `DynamoDbWaiter` before attempting to print out the table name received in the response. The creation of a table takes some time. Therefore, using a waiter means you don't have to write logic that polls the DynamoDB service to see if the table exists before using the table.

Imports

```
import com.example.dynamodb.Customer;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

Code

```
public static void createCustomerTable(DynamoDbTable<Customer> customerTable,
DynamoDbClient standardClient) {
    // Create the DynamoDB table using the 'customerTable' DynamoDbTable instance.
    customerTable.createTable(builder -> builder
            .provisionedThroughput(b -> b
                    .readCapacityUnits(10L)
                    .writeCapacityUnits(10L)
                    .build())
    );
    // The DynamoDbClient instance (named 'standardClient') passed to the builder for
the DynamoDbWaiter is the same instance
    // that was passed to the builder of the DynamoDbEnhancedClient instance that we
created previously.
    // By using the same instance, it ensures that the same Region that was configured
on the standard DynamoDbClient
    // instance is used for other service clients that accept a DynamoDbClient during
construction.
```

```
    try (DynamoDbWaiter waiter =
DynamoDbWaiter.builder().client(standardClient).build()) { // DynamoDbWaiter is
Autocloseable
        ResponseOrException<DescribeTableResponse> response = waiter
                .waitUntilTableExists(builder ->
builder.tableName("Customer").build())
                .matched();
        DescribeTableResponse tableDescription = response.response().orElseThrow(
                () -> new RuntimeException("Customer table was not created."));
        // The actual error can be inspected in response.exception()
        logger.info("Customer table was created.");
    }
}
```

Note

A DynamoDB table's attribute names begin with a lowercase letter when the table is generated from a data class. If you want the table's attribute name to begin with an uppercase letter, use the `@DynamoDbAttribute(NAME)` annotation and provide the name you want as a parameter.

Perform operations

After the table is created, use the `DynamoDbTable` instance to perform operations against the DynamoDB table.

In the following example, a singleton `DynamoDbTable<Customer>` is passed as a parameter along with a `Customer` data class instance to add a new item to the table.

```
   public static void putItemExample(DynamoDbTable<Customer> customerTable, Customer
customer){
       logger.info(customer.toString());
       customerTable.putItem(customer);
   }
```

`Customer` object

```
       Customer customer = new Customer();
       customer.setId("1");
```

```
        customer.setCustName("Customer Name");
        customer.setEmail("customer@example.com");
        customer.setRegistrationDate(Instant.parse("2023-07-03T10:15:30.00Z"));
```

Before sending the `customer` object to the DynamoDB service, log the output of the object's `toString()` method to compare it to what the enhanced client sends.

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

Wire-level logging shows the payload of the generated request. The enhanced client generated the low-level representation from the data class. The `regDate` attribute, which is an `Instant` type in Java, is represented as a DynamoDB string.


```
{
  "TableName": "Customer",
  "Item": {
    "registrationDate": {
      "S": "2023-07-03T10:15:30Z"
    },
    "id": {
      "S": "1"
    },
    "custName": {
      "S": "Customer Name"
    },
    "email": {
      "S": "customer@example.com"
    }
  }
}
```


Work with an existing table

The previous section showed how to create a DynamoDB table starting with a Java data class. If you already have an existing table and want to use the features of the enhanced client, you can create a Java data class to work with the table. You need to examine the DynamoDB table and add the necessary annotations to the data class.

Before you work with an existing table, call the `DynamoDbEnhanced.table()` method. This was done in the previous example with the following statement.

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

After the `DynamoDbTable` instance is returned, you can begin working right away with the underlying table. You do not need to recreate the table by calling the `DynamoDbTable.createTable()` method.

The following example demonstrates this by immediately retrieving a `Customer` instance from the DynamoDB table.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",
 TableSchema.fromBean(Customer.class));
// The Customer table exists already and has an item with a primary key value of "1"
 and a sort key value of "customer@example.com".
customerTable.getItem(
        Key.builder().
                partitionValue("1").
                sortValue("customer@example.com").build());
```

> **Important**
>
> The table name used in the `table()` method must match the existing DynamoDB table name.

Learn the basics of the DynamoDB Enhanced Client API

This topic discusses the basic features of the DynamoDB Enhanced Client API and compares it to the standard DynamoDB client API.

If you are new to the DynamoDB Enhanced Client API, we recommend that you go through the introductory tutorial to familiarize yourself with fundamental classes.

DynamoDB items in Java

DynamoDB tables store items. Depending on your use case, items on the Java side can take the form of statically structured data or structures created dynamically.

If your use case calls for items with a consistent set of attributes, use annotated classes or use a builder to generate the appropriate statically-typed `TableSchema`.

Alternatively, if you need to store items that consist of varying structures, create a `DocumentTableSchema`. `DocumentTableSchema` is part of the Enhanced Document API and requires only a statically-typed primary key and works with `EnhancedDocument` instances to hold the data elements. The Enhanced Document API is covered in another topic.

Attribute types for data model classes

Although DynamoDB supports a small number of attribute types compared to the rich type system of Java, the DynamoDB Enhanced Client API provides mechanisms to convert members of a Java class to and from DynamoDB attribute types.

The attribute types (properties) of your Java data classes should be objects types, not primitives. For example, always use `Long` and `Integer` object data types, not `long` and `int` primitives.

By default, the DynamoDB Enhanced Client API supports attribute converters for a large number of types, such as Integer, String, BigDecimal, and Instant. The list appears in the known implementing classes of the AttributeConverter interface. The list includes many types and collections such as maps, lists, and sets.

To store the data for an attribute type that isn't supported by default or doesn't conform to the JavaBean convention, you can write a custom `AttributeConverter` implementation to do the conversion. See the attribute conversion section for an example.

To store the data for an attribute type whose class conforms to the Java beans specification (or an immutable data class), you can take two approaches.

- If you have access to the source file, you can annotate the class with `@DynamoDbBean` (or `@DynamoDbImmutable`). The section that discusses nested attributes shows examples of using annotated classes.

- If do not have access to the source file of the JavaBean data class for the attribute (or you don't want to annotate the source file of a class that you do have access to), then you can use the builder approach. This creates a table schema without defining the keys. Then, you can nest this table schema inside another table schema to perform the mapping. The nested attribute section has an example showing use of nested schemas.

Null values

When you use the `putItem` API, the enhanced client does not include null-valued attributes of a mapped data object in the request to DynamoDB.

For `updateItem` requests, null-valued attributes are removed from the item on the database. If you intend to update some attribute values and keep the other unchanged, either copy the values of other attributes that should not be changed or use the ignoreNull() method on the update builder.

The following example demonstrates `ignoreNulls()` for `the updateItem()` method.

```
    public void updateItemNullsExample(){
        Customer customer = new Customer();
        customer.setCustName("CustName");
        customer.setEmail("email");
        customer.setId("1");
        customer.setRegistrationDate(Instant.now());

        // Put item with values for all attributes.
        customerDynamoDbTable.putItem(customer);

        // Create a Customer instance with the same id value, but a different name
 value.
        // Do not set the 'registrationDate' attribute.
        Customer custForUpdate = new Customer();
        custForUpdate.setCustName("NewName");
        custForUpdate.setEmail("email");
        custForUpdate.setId("1");

        // Update item without setting the registrationDate attribute.
        customerDynamoDbTable.updateItem(b -> b
                .item(custForUpdate)
                .ignoreNulls(Boolean.TRUE));

        Customer updatedWithNullsIgnored = customerDynamoDbTable.getItem(customer);
        // registrationDate value is unchanged.
        logger.info(updatedWithNullsIgnored.toString());

        customerDynamoDbTable.updateItem(custForUpdate);
        Customer updatedWithNulls = customerDynamoDbTable.getItem(customer);
        // registrationDate value is null because ignoreNulls() was not used.
        logger.info(updatedWithNulls.toString());
```

```
    }
}

// Logged lines.
Customer [id=1, custName=NewName, email=email,
 registrationDate=2023-04-05T16:32:32.056Z]
Customer [id=1, custName=NewName, email=email, registrationDate=null]
```

DynamoDB Enhanced Client basic methods

Basic methods of the enhanced client map to the DynamoDB service operations that they're named after. The following examples show the simplest variation of each method. You can customize each method by passing in an enhanced request object. Enhanced request objects offer most of the features available in the standard DynamoDB client. They are fully documented in the AWS SDK for Java 2.x API Reference.

The example uses the the section called "`Customer` class" shown previously.

```
// CreateTable
customerTable.createTable();

// GetItem
Customer customer =
 customerTable.getItem(Key.builder().partitionValue("a123").build());

// UpdateItem
Customer updatedCustomer = customerTable.updateItem(customer);

// PutItem
customerTable.putItem(customer);

// DeleteItem
Customer deletedCustomer =
 customerTable.deleteItem(Key.builder().partitionValue("a123").sortValue(456).build());

// Query
PageIterable<Customer> customers = customerTable.query(keyEqualTo(k ->
 k.partitionValue("a123")));

// Scan
PageIterable<Customer> customers = customerTable.scan();

// BatchGetItem
```

```
BatchGetResultPageIterable batchResults =
    enhancedClient.batchGetItem(r -> r.addReadBatch(ReadBatch.builder(Customer.class)
                                       .mappedTableResource(customerTable)
                                       .addGetItem(key1)
                                       .addGetItem(key2)
                                       .addGetItem(key3)
                                       .build()));

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

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

// TransactWriteItems
enhancedClient.transactWriteItems(r -> r.addConditionCheck(customerTable,
                                                            i -> i.key(orderKey)

 .conditionExpression(conditionExpression))
                                         .addUpdateItem(customerTable, customer)
                                         .addDeleteItem(customerTable, key));
```

Compare DynamoDB Enhanced Client to standard DynamoDB client

Both DynamoDB client APIs—standard and enhanced—let you work with DynamoDB tables to perform CRUD (create, read, update and delete) data-level operations. The difference between the client APIs is how that is accomplished. Using the standard client, you work directly with low-level data attributes. The enhanced client API uses familiar Java classes and maps to the low-level API behind the scenes.

While both client APIs support data-level operations, the standard DynamoDB client also supports resource-level operations. Resource-level operations manage the database, such as creating backups, listing tables, and updating tables. The enhanced client API supports a select number of resource-level operations such as creating, describing, and deleting tables.

To illustrate the different approaches used by the two client APIs, the following code examples show the creation of the same `ProductCatalog` table using the standard client and the enhanced client.

Compare: Create a table using the standard DynamoDB client

```
DependencyFactory.dynamoDbClient().createTable(builder -> builder
        .tableName(TABLE_NAME)
        .attributeDefinitions(
                b -> b.attributeName("id").attributeType(ScalarAttributeType.N),
                b -> b.attributeName("title").attributeType(ScalarAttributeType.S),
                b -> b.attributeName("isbn").attributeType(ScalarAttributeType.S)
        )
        .keySchema(
                builder1 -> builder1.attributeName("id").keyType(KeyType.HASH),
                builder2 -> builder2.attributeName("title").keyType(KeyType.RANGE)
        )
        .globalSecondaryIndexes(builder3 -> builder3
                        .indexName("products_by_isbn")
                        .keySchema(builder2 -> builder2
                                .attributeName("isbn").keyType(KeyType.HASH))
                        .projection(builder2 -> builder2
                                .projectionType(ProjectionType.INCLUDE)
                                .nonKeyAttributes("price", "authors"))
                        .provisionedThroughput(builder4 -> builder4
                                .writeCapacityUnits(5L).readCapacityUnits(5L))
        )
        .provisionedThroughput(builder1 -> builder1
                .readCapacityUnits(5L).writeCapacityUnits(5L))
);
```

Compare: Create a table using the DynamoDB Enhanced Client

```
DynamoDbEnhancedClient enhancedClient = DependencyFactory.enhancedClient();
productCatalog = enhancedClient.table(TABLE_NAME,
 TableSchema.fromImmutableClass(ProductCatalog.class));
productCatalog.createTable(b -> b
        .provisionedThroughput(b1 -> b1.readCapacityUnits(5L).writeCapacityUnits(5L))
        .globalSecondaryIndices(b2 -> b2.indexName("products_by_isbn")
                .projection(b4 -> b4
```

```
                        .projectionType(ProjectionType.INCLUDE)
                        .nonKeyAttributes("price", "authors"))
                .provisionedThroughput(b3 ->
 b3.writeCapacityUnits(5L).readCapacityUnits(5L))
        )
);
```

The enhanced client uses the following annotated data class. The DynamoDB Enhanced Client maps Java data types to DynamoDB data types for less verbose code that is easier to follow. `ProductCatalog` is an example of using an immutable class with the DynamoDB Enhanced Client. The use of Immutable classes for mapped data classes is discussed later in this topic.

`ProductCatalog` class

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbIgnore;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.math.BigDecimal;
import java.util.Objects;
import java.util.Set;

@DynamoDbImmutable(builder = ProductCatalog.Builder.class)
public class ProductCatalog implements Comparable<ProductCatalog> {
    private Integer id;
    private String title;
    private String isbn;
    private Set<String> authors;
    private BigDecimal price;


    private ProductCatalog(Builder builder){
        this.authors = builder.authors;
        this.id = builder.id;
        this.isbn = builder.isbn;
        this.price = builder.price;
        this.title = builder.title;
```

```
    }

    public static Builder builder(){ return new Builder(); }

    @DynamoDbPartitionKey
    public Integer id() { return id; }

    @DynamoDbSortKey
    public String title() { return title; }

    @DynamoDbSecondaryPartitionKey(indexNames = "products_by_isbn")
    public String isbn() { return isbn; }
    public Set<String> authors() { return authors; }
    public BigDecimal price() { return price; }


    public static final class Builder {
      private Integer id;
      private String title;
      private String isbn;
      private Set<String> authors;
      private BigDecimal price;
      private Builder(){}

      public Builder id(Integer id) { this.id = id; return this; }
      public Builder title(String title) { this.title = title; return this; }
      public Builder isbn(String ISBN) { this.isbn = ISBN; return this; }
      public Builder authors(Set<String> authors) { this.authors = authors; return
 this; }
      public Builder price(BigDecimal price) { this.price = price; return this; }
      public ProductCatalog build() { return new ProductCatalog(this); }
  }

    @Override
    public String toString() {
        final StringBuffer sb = new StringBuffer("ProductCatalog{");
        sb.append("id=").append(id);
        sb.append(", title='").append(title).append('\'');
        sb.append(", isbn='").append(isbn).append('\'');
        sb.append(", authors=").append(authors);
        sb.append(", price=").append(price);
        sb.append('}');
        return sb.toString();
    }
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        ProductCatalog that = (ProductCatalog) o;
        return id.equals(that.id) && title.equals(that.title) && Objects.equals(isbn,
 that.isbn) && Objects.equals(authors, that.authors) && Objects.equals(price,
 that.price);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, title, isbn, authors, price);
    }

    @Override
    @DynamoDbIgnore
    public int compareTo(ProductCatalog other) {
        if (this.id.compareTo(other.id) != 0){
            return this.id.compareTo(other.id);
        } else {
            return this.title.compareTo(other.title);
        }
    }
}
```

The following two code examples of a batch write illustrate the verboseness and lack of type safety when using the standard client as opposed to the enhanced client.

Compare: Batch write using the standard DynamoDB client

```
    public static void batchWriteStandard(DynamoDbClient dynamoDbClient, String
 tableName) {

        Map<String, AttributeValue> catalogItem = Map.of(
                "authors", AttributeValue.builder().ss("a", "b").build(),
                "id", AttributeValue.builder().n("1").build(),
                "isbn", AttributeValue.builder().s("1-565-85698").build(),
                "title", AttributeValue.builder().s("Title 1").build(),
                "price", AttributeValue.builder().n("52.13").build());

        Map<String, AttributeValue> catalogItem2 = Map.of(
```

```
                "authors", AttributeValue.builder().ss("a", "b", "c").build(),
                "id", AttributeValue.builder().n("2").build(),
                "isbn", AttributeValue.builder().s("1-208-98073").build(),
                "title", AttributeValue.builder().s("Title 2").build(),
                "price", AttributeValue.builder().n("21.99").build());

        Map<String, AttributeValue> catalogItem3 = Map.of(
                "authors", AttributeValue.builder().ss("g", "k", "c").build(),
                "id", AttributeValue.builder().n("3").build(),
                "isbn", AttributeValue.builder().s("7-236-98618").build(),
                "title", AttributeValue.builder().s("Title 3").build(),
                "price", AttributeValue.builder().n("42.00").build());

        Set<WriteRequest> writeRequests = Set.of(
                WriteRequest.builder().putRequest(b -> b.item(catalogItem)).build(),
                WriteRequest.builder().putRequest(b -> b.item(catalogItem2)).build(),
                WriteRequest.builder().putRequest(b -> b.item(catalogItem3)).build());

        Map<String, Set<WriteRequest>> productCatalogItems = Map.of(
                "ProductCatalog", writeRequests);

        BatchWriteItemResponse response = dynamoDbClient.batchWriteItem(b ->
 b.requestItems(productCatalogItems));

        logger.info("Unprocessed items: " + response.unprocessedItems().size());
    }
```

Compare: Batch write using the DynamoDB Enhanced Client

```
    public static void batchWriteEnhanced(DynamoDbTable<ProductCatalog> productCatalog)
 {
        ProductCatalog prod = ProductCatalog.builder()
                .id(1)
                .isbn("1-565-85698")
                .authors(new HashSet<>(Arrays.asList("a", "b")))
                .price(BigDecimal.valueOf(52.13))
                .title("Title 1")
                .build();
        ProductCatalog prod2 = ProductCatalog.builder()
                .id(2)
                .isbn("1-208-98073")
                .authors(new HashSet<>(Arrays.asList("a", "b", "c")))
                .price(BigDecimal.valueOf(21.99))
```

```
                .title("Title 2")
                .build();
        ProductCatalog prod3 = ProductCatalog.builder()
                .id(3)
                .isbn("7-236-98618")
                .authors(new HashSet<>(Arrays.asList("g", "k", "c")))
                .price(BigDecimal.valueOf(42.00))
                .title("Title 3")
                .build();

        BatchWriteResult batchWriteResult = DependencyFactory.enhancedClient()
                .batchWriteItem(b -> b.writeBatches(
                        WriteBatch.builder(ProductCatalog.class)
                                .mappedTableResource(productCatalog)
                                .addPutItem(prod).addPutItem(prod2).addPutItem(prod3)
                                .build()
                ));
        logger.info("Unprocessed items: " +
 batchWriteResult.unprocessedPutItemsForTable(productCatalog).size());
    }
```

Work with immutable data classes

The mapping feature of the DynamoDB Enhanced Client API works with immutable data classes. An immutable class has only getters and requires a builder class that the SDK uses to create instances of the class. Instead of using the `@DynamoDbBean` annotation as shown in the [Customer class](), immutable classes use the `@DynamoDbImmutable` annotation, which takes a parameter that indicates the builder class to use.

The following class is an immutable version of `Customer`.

```
package org.example.tests.model.immutable;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.time.Instant;
```

```
@DynamoDbImmutable(builder = CustomerImmutable.Builder.class)
public class CustomerImmutable {
    private final String id;
    private final String name;
    private final String email;
    private final Instant regDate;

    private CustomerImmutable(Builder b) {
        this.id = b.id;
        this.email = b.email;
        this.name = b.name;
        this.regDate = b.regDate;
    }

    // This method will be automatically discovered and used by the TableSchema.
    public static Builder builder() { return new Builder(); }

    @DynamoDbPartitionKey
    public String id() { return this.id; }

    @DynamoDbSortKey
    public String email() { return this.email; }

    @DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name")
    public String name() { return this.name; }

    @DynamoDbSecondarySortKey(indexNames = {"customers_by_date", "customers_by_name"})
    public Instant regDate() { return this.regDate; }

    public static final class Builder {
        private String id;
        private String email;
        private String name;
        private Instant regDate;

        // The private Builder constructor is visible to the enclosing
 CustomerImmutable class.
        private Builder() {}

        public Builder id(String id) { this.id = id; return this; }
        public Builder email(String email) { this.email = email; return this; }
        public Builder name(String name) { this.name = name; return this; }
```

```
        public Builder regDate(Instant regDate) { this.regDate = regDate; return
 this; }

        // This method will be automatically discovered and used by the TableSchema.
        public CustomerImmutable build() { return new CustomerImmutable(this); }
    }
}
```

You must meet the following requirements when you annotate a data class with `@DynamoDbImmutable`.

1. Every method that is both not an override of `Object.class` and has not been annotated with `@DynamoDbIgnore` must be a getter for an attribute of the DynamoDB table.
2. Every getter must have a corresponding case-sensitive setter on the builder class.
3. Only one of the following construction conditions must be met.
   - The builder class must have a public default constructor.
   - The data class must have a public static method named `builder()` that takes no parameters and returns an instance of the builder class. This option is shown in the immutable `Customer` class.
4. The builder class must have a public method named `build()` that takes no parameters and returns an instance of the immutable class.

To create a `TableSchema` for your immutable class, use the `fromImmutableClass()` method on `TableSchema` as shown in the following snippet.

```
static final TableSchema<CustomerImmutable> customerImmutableTableSchema =
                         TableSchema.fromImmutableClass(CustomerImmutable.class);
```

Just as you can create a DynamoDB table from a mutable class, you can create one from an immutable class with a *one-time* call to `createTable()` of `DynamoDbTable` as shown in the following snippet example.

```
static void createTableFromImmutable(DynamoDbEnhancedClient enhancedClient, String
 tableName, DynamoDbWaiter waiter){
    // First, create an in-memory representation of the table using the 'table()'
 method of the DynamoDb Enhanced Client.
    // 'table()' accepts a name for the table and a TableSchema instance that you
 created previously.
```

```
    DynamoDbTable<CustomerImmutable> customerDynamoDbTable = enhancedClient
            .table(tableName, TableSchema.fromImmutableClass(CustomerImmutable.class));

    // Second, call the 'createTable()' method on the DynamoDbTable instance.
    customerDynamoDbTable.createTable();
    waiter.waitUntilTableExists(b -> b.tableName(tableName));
}
```

Use third-party libraries, such as Lombok

Third-party libraries, such as Project Lombok, help generate boilerplate code associated with immutable objects. The DynamoDB Enhanced Client API works with these libraries as long as the data classes follow the conventions detailed in this section.

The following example shows the immutable `CustomerImmutable` class with Lombok annotations. Note how Lombok's `onMethod` feature copies attribute-based DynamoDB annotations, such as `@DynamoDbPartitionKey`, onto the generated code.

```
@Value
@Builder
@DynamoDbImmutable(builder = Customer.CustomerBuilder.class)
public class Customer {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;

    @Getter(onMethod_=@DynamoDbSortKey)
    private String email;

    @Getter(onMethod_=@DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name"))
    private String name;

    @Getter(onMethod_=@DynamoDbSecondarySortKey(indexNames = {"customers_by_date",
 "customers_by_name"}))
    private Instant createdDate;
}
```

Use expressions and conditions

Expressions in the DynamoDB Enhanced Client API are Java representations of DynamoDB expressions.

The DynamoDB Enhanced Client API uses three types of expressions:

Expression

The `Expression` class is used when you define conditions and filters.

`QueryConditional`

This type of expression represents key conditions for query operations.

`UpdateExpression`

This class helps you write DynamoDB update expressions and is currently used in the extension framework when you update an item.

Expression anatomy

An expression is made up of the following:

- A string expression (required). The string contains a DynamoDB logic expression with placeholder names for attribute names and attribute values.
- A map of expression values (usually required).
- A map of expression names (optional).

Use a builder to generate an`Expression` object that takes the following general form.

```
Expression expression = Expression.builder()
                            .expression(<String>)
                            .expressionNames(<Map>)
                            .expressionValues(<Map>)
                           .build()
```

`Expressions` usually require a map of expression values. The map provides the values for the placeholders in the string expression. The map key consists of the placeholder name preceded with a colon (`:`) and the map value is an instance of AttributeValue. The AttributeValues class has convenience methods to generate an `AttributeValue` instance from a literal. Alternatively, you can use the `AttributeValue.Builder` to generate an `AttributeValue` instance.

The following snippet shows a map with two entries after comment line 2. The string passed to the `expression()` method, shown after comment line 1, contains the placeholders that DynamoDB resolves before performing the operation. This snippet doesn't contain a map of expression names, because *price* is a permissible attribute name.

```
    public static void scanAsync(DynamoDbAsyncTable productCatalog) {
        ScanEnhancedRequest request = ScanEnhancedRequest.builder()
                .consistentRead(true)
                .attributesToProject("id", "title", "authors", "price")
                .filterExpression(Expression.builder()
                        // 1. :min_value and :max_value are placeholders for the values
 provided by the map
                        .expression("price >= :min_value AND price <= :max_value")
                        // 2. Two values are needed for the expression and each is
 supplied as a map entry.
                        .expressionValues(
                                Map.of( ":min_value", numberValue(8.00),
                                        ":max_value", numberValue(400_000.00)))
                        .build())
                .build();
```

If an attribute name in the DynamoDB table is a reserved word, begins with a number, or contains a space, a map of expression names is required for the `Expression`.

For example, if the attribute name was *`1price`* instead of *`price`* in the previous code example, the example would need to be modified as shown in the following example.

```
        ScanEnhancedRequest request = ScanEnhancedRequest.builder()
                .filterExpression(Expression.builder()
                        .expression("#price >= :min_value AND #price <= :max_value")
                        .expressionNames( Map.of("#price", "1price") )
                        .expressionValues(
                                Map.of(":min_value", numberValue(8.00),
                                        ":max_value", numberValue(400_000.00)))
                        .build())
                .build();
```

A placeholder for an expression name begins with the pound sign (#). An entry for the map of expression names uses the placeholder as the key and the attribute name as the value. The map is added to the expression builder with the `expressionNames()` method. DynamoDB resolves the attribute name before it performs the operation.

Expression values are not required if a function is used in the string expression. An example of an expression function is `attribute_exists(<attribute_name>)`.

The following example builds an `Expression` that uses a DynamoDB function. The expression string in this example uses no placeholders. This expression could be used on a `putItem` operation to check if an item already exists in the database with a `movie` attribute's value equal to the data object's `movie` attribute.

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

The DynamoDB Developer Guide contains complete information on the low-level expressions that are used with DynamoDB.

Condition expressions and conditionals

When you use the `putItem()`, `updateItem()`, and `deleteItem()` methods, and also when you use transaction and batch operations, you use `Expression` objects to specify conditions that DynamoDB must meet to proceed with the operation. These expressions are named condition expressions. For an example, see the condition expression used in the `addDeleteItem()` method (after comment line 1) of transaction example shown in this guide.

When you work with the `query()` methods, a condition is expressed as a `QueryConditional`. The `QueryConditional` class has several static convenience methods that help you write the criteria that determine which items to read from DynamoDB.

For examples of `QueryConditionals`, see the first code example of the the section called “Query method examples” section of this guide.

Filter expressions

Filter expressions are used in scan and query operations to filter the items that are returned.

A filter expression is applied after all the data is read from the database, so the read cost is the same as if there were no filter. The *Amazon DynamoDB Developer Guide* has more information about using filter expressions for both query and scan operations.

The following example shows a filter expression added to a scan request. The criteria restricts the items returned to items with a price between 8.00 and 80.00 inclusive.

```
        Map<String, AttributeValue> expressionValues = Map.of(
                ":min_value", numberValue(8.00),
```

```
                ":max_value", numberValue(80.00));

        ScanEnhancedRequest request = ScanEnhancedRequest.builder()
                .consistentRead(true)
                // 1. the 'attributesToProject()' method allows you to specify which
 values you want returned.
                .attributesToProject("id", "title", "authors", "price")
                // 2. Filter expression limits the items returned that match the
 provided criteria.
                .filterExpression(Expression.builder()
                        .expression("price >= :min_value AND price <= :max_value")
                        .expressionValues(expressionValues)
                        .build())
                .build();
```

Update expressions

The DynamoDB Enhanced Client's `updateItem()` method provides a standard way to update items in DynamoDB. However, when you require more functionality, UpdateExpressions provide a type-safe representation of DynamoDB update expression syntax. For example, you can use `UpdateExpressions` to increase values without first reading items from DynamoDB, or add individual members to a list. Update expressions are currently available in custom extensions for the `updateItem()` method.

For an example that uses update expressions, see the custom extension example in this guide.

More information about update expressions is available in the Amazon DynamoDB Developer Guide.

Work with paginated results: scans and queries

The `scan`, `query` and `batch` methods of the DynamoDB Enhanced Client API return responses with one or more *pages*. A page contains one or more items. Your code can process the response on per-page basis or it can process individual items.

A paginated response returned by the synchronous `DynamoDbEnhancedClient` client returns a PageIterable object, whereas a response returned by the asynchronous `DynamoDbEnhancedAsyncClient` returns a PagePublisher object.

This section looks at processing paginated results and provides examples that use the scan and query APIs.

Scan a table

The SDK's scan method corresponds to the DynamoDB operation of the same name. The DynamoDB Enhanced Client API offers the same options but it uses a familiar object model and handles the pagination for you.

First, we explore the `PageIterable` interface by looking at the `scan` method of the synchronous mapping class, DynamoDbTable.

Use the synchronous API

The following example shows the `scan` method that uses an expression to filter the items that are returned. The ProductCatalog is the model object that was shown earlier.

The filtering expression shown after comment line 2 limits the `ProductCatalog` items that are returned to those with a price value between 8.00 and 80.00 inclusively.

This example also excludes the `isbn` values by using the `attributesToProject` method shown after comment line 1.

After comment line 3, the `PageIterable` object, `pagedResults`, is returned by the `scan` method. The `stream` method of `PageIterable` returns a `java.util.Stream` object, which you can use to process the pages. In this example, the number of pages is counted and logged.

Starting with comment line 4, the example shows two variations of accessing the `ProductCatalog` items. The version after comment line 4a streams through each page and sorts and logs the items on each page. The version after comment line 4b skips the page iteration and accesses the items directly.

The `PageIterable` interface offers multiple ways to process results because of its two parent interfaces—`java.lang.Iterable` and `SdkIterable`. `Iterable` brings the `forEach`, `iterator` and `spliterator` methods, and `SdkIterable` brings the `stream` method.

```
    public static void scanSync(DynamoDbTable<ProductCatalog> productCatalog) {

        Map<String, AttributeValue> expressionValues = Map.of(
                ":min_value", numberValue(8.00),
                ":max_value", numberValue(80.00));

        ScanEnhancedRequest request = ScanEnhancedRequest.builder()
                .consistentRead(true)
```

```
                // 1. the 'attributesToProject()' method allows you to specify which
 values you want returned.
                .attributesToProject("id", "title", "authors", "price")
                // 2. Filter expression limits the items returned that match the
 provided criteria.
                .filterExpression(Expression.builder()
                        .expression("price >= :min_value AND price <= :max_value")
                        .expressionValues(expressionValues)
                        .build())
                .build();

        // 3. A PageIterable object is returned by the scan method.
        PageIterable<ProductCatalog> pagedResults = productCatalog.scan(request);
        logger.info("page count: {}", pagedResults.stream().count());

        // 4. Log the returned ProductCatalog items using two variations.
        // 4a. This version sorts and logs the items of each page.
        pagedResults.stream().forEach(p -> p.items().stream()
                .sorted(Comparator.comparing(ProductCatalog::price))
                .forEach(
                        item -> logger.info(item.toString())
                ));
        // 4b. This version sorts and logs all items for all pages.
        pagedResults.items().stream()
                .sorted(Comparator.comparing(ProductCatalog::price))
                .forEach(
                        item -> logger.info(item.toString())
                );
    }
```

Use the asynchronous API

The asynchronous `scan` method returns results as a `PagePublisher` object. The `PagePublisher` interface has two `subscribe` methods that you can use to process response pages. One `subscribe` method comes from the `org.reactivestreams.Publisher` parent interface. To process pages using this first option, pass a [`Subscriber`] instance to the `subscribe` method. The first example that follows shows the use of `subscribe` method.

The second `subscribe` method comes from the [SdkPublisher] interface. This version of `subscribe` accepts a [`Consumer`] rather than a `Subscriber`. This `subscribe` method variation is shown in the second example that follows.

The following example shows the asynchronous version of the `scan` method that uses the same filter expression shown in the previous example.

After comment line 3, `DynamoDbAsyncTable.scan` returns a `PagePublisher` object. On the next line, the code creates an instance of the `org.reactivestreams.Subscriber` interface, `ProductCatalogSubscriber`, which subscribes to the `PagePublisher` after comment line 4.

The `Subscriber` object collects the `ProductCatalog` items from each page in the `onNext` method after comment line 8 in the `ProductCatalogSubscriber` class example. The items are stored in the private `List` variable and are accessed in the calling code with the `ProductCatalogSubscriber.getSubscribedItems()` method. This is called after comment line 5.

After the list is retrieved, the code sorts all `ProductCatalog` items by price and logs each item.

The [CountDownLatch](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CountDownLatch.html) in the `ProductCatalogSubscriber` class blocks the calling thread until all items have been added to the list before continuing after comment line 5.

```
    public static void scanAsync(DynamoDbAsyncTable productCatalog) {
        ScanEnhancedRequest request = ScanEnhancedRequest.builder()
                .consistentRead(true)
                .attributesToProject("id", "title", "authors", "price")
                .filterExpression(Expression.builder()
                        // 1. :min_value and :max_value are placeholders for the values
 provided by the map
                        .expression("price >= :min_value AND price <= :max_value")
                        // 2. Two values are needed for the expression and each is
 supplied as a map entry.
                        .expressionValues(
                                Map.of( ":min_value", numberValue(8.00),
                                        ":max_value", numberValue(400_000.00)))
                        .build())
                .build();

        // 3. A PagePublisher object is returned by the scan method.
        PagePublisher<ProductCatalog> pagePublisher = productCatalog.scan(request);
        ProductCatalogSubscriber subscriber = new ProductCatalogSubscriber();
        // 4. Subscribe the ProductCatalogSubscriber to the PagePublisher.
        pagePublisher.subscribe(subscriber);
        // 5. Retrieve all collected ProductCatalog items accumulated by the
 subscriber.
        subscriber.getSubscribedItems().stream()
```

```
                .sorted(Comparator.comparing(ProductCatalog::price))
                .forEach(item ->
                        logger.info(item.toString()));
        // 6. Use a Consumer to work through each page.
        pagePublisher.subscribe(page -> page
                        .items().stream()
                        .sorted(Comparator.comparing(ProductCatalog::price))
                        .forEach(item ->
                                logger.info(item.toString())))
                .join(); // If needed, blocks the subscribe() method thread until it is
 finished processing.
        // 7. Use a Consumer to work through each ProductCatalog item.
        pagePublisher.items()
                .subscribe(product -> logger.info(product.toString()))
                .exceptionally(failure -> {
                    logger.error("ERROR  - ", failure);
                    return null;
                })
                .join(); // If needed, blocks the subscribe() method thread until it is
 finished processing.
    }
```

```
    private static class ProductCatalogSubscriber implements
 Subscriber<Page<ProductCatalog>> {
        private CountDownLatch latch = new CountDownLatch(1);
        private Subscription subscription;
        private List<ProductCatalog> itemsFromAllPages = new ArrayList<>();

        @Override
        public void onSubscribe(Subscription sub) {
            subscription = sub;
            subscription.request(1L);
            try {
                latch.await(); // Called by main thread blocking it until latch is
 released.
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }

        @Override
        public void onNext(Page<ProductCatalog> productCatalogPage) {
```

```
            // 8. Collect all the ProductCatalog instances in the page, then ask the
 publisher for one more page.
            itemsFromAllPages.addAll(productCatalogPage.items());
            subscription.request(1L);
        }

        @Override
        public void onError(Throwable throwable) {
        }

        @Override
        public void onComplete() {
            latch.countDown(); // Call by subscription thread; latch releases.
        }

        List<ProductCatalog> getSubscribedItems() {
            return this.itemsFromAllPages;
        }
    }
```

The following snippet example uses the version of the `PagePublisher.subscribe` method that accepts a `Consumer` after comment line 6. The Java lambda parameter consumes pages, which further process each item. In this example, each page is processed and the items on each page are sorted and then logged.

```
        // 6. Use a Consumer to work through each page.
        pagePublisher.subscribe(page -> page
                        .items().stream()
                        .sorted(Comparator.comparing(ProductCatalog::price))
                        .forEach(item ->
                                logger.info(item.toString())))
                .join(); // If needed, blocks the subscribe() method thread until it is
 finished processing.
```

The `items` method of `PagePublisher` unwraps the model instances so that your code can process the items directly. This approach is shown in the following snippet.

```
        // 7. Use a Consumer to work through each ProductCatalog item.
        pagePublisher.items()
                .subscribe(product -> logger.info(product.toString()))
                .exceptionally(failure -> {
                    logger.error("ERROR  - ", failure);
```

```
                    return null;
                })
                .join(); // If needed, blocks the subscribe() method thread until it is
 finished processing.
```

Query a table

The `query()` method of the `DynamoDbTable` class finds items based on primary key values. The `@DynamoDbPartitionKey` annotation and the optional `@DynamoDbSortKey` annotation are used to define the primary key on your data class.

The `query()` method requires a partition key value that finds items that match the supplied value. If your table also defines a sort key, you can add a value for it to your query as an additional comparison condition to fine tune the results.

Except for processing the results, the synchronous and asynchronous versions of `query()` work the same. As with the `scan` API, the `query` API returns a `PageIterable` for a synchronous call and a `PagePublisher` for asynchronous call. We discussed the use of `PageIterable` and `PagePublisher` previously in the scan section.

Query method examples

The `query()` method code example that follows uses the `MovieActor` class. The data class defines a composite primary key that is made up of the **`movie`** attribute for the partition key and the **`actor`** attribute for the sort key.

The class also signals that it uses a global secondary index named **`acting_award_year`**. The index's composite primary key is composed of the **`actingaward`** attribute for the partition key and the **`actingyear`** for the sort key. Later in this topic, when we show how to create and use indexes, we'll refer to the **`acting_award_year`** index.

`MovieActor` class

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbAttribute;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
```

```
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.util.Objects;

@DynamoDbBean
public class MovieActor implements Comparable<MovieActor> {

    private String movieName;
    private String actorName;
    private String actingAward;
    private Integer actingYear;
    private String actingSchoolName;

    @DynamoDbPartitionKey
    @DynamoDbAttribute("movie")
    public String getMovieName() {
        return movieName;
    }

    public void setMovieName(String movieName) {
        this.movieName = movieName;
    }

    @DynamoDbSortKey
    @DynamoDbAttribute("actor")
    public String getActorName() {
        return actorName;
    }

    public void setActorName(String actorName) {
        this.actorName = actorName;
    }

    @DynamoDbSecondaryPartitionKey(indexNames = "acting_award_year")
    @DynamoDbAttribute("actingaward")
    public String getActingAward() {
        return actingAward;
    }

    public void setActingAward(String actingAward) {
        this.actingAward = actingAward;
    }
```

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

    public void setActingYear(Integer actingYear) {
        this.actingYear = actingYear;
    }

    @DynamoDbAttribute("actingschoolname")
    public String getActingSchoolName() {
        return actingSchoolName;
    }

    public void setActingSchoolName(String actingSchoolName) {
        this.actingSchoolName = actingSchoolName;
    }

    @Override
    public String toString() {
        final StringBuffer sb = new StringBuffer("MovieActor{");
        sb.append("movieName='").append(movieName).append('\'');
        sb.append(", actorName='").append(actorName).append('\'');
        sb.append(", actingAward='").append(actingAward).append('\'');
        sb.append(", actingYear=").append(actingYear);
        sb.append(", actingSchoolName='").append(actingSchoolName).append('\'');
        sb.append('}');
        return sb.toString();
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        MovieActor that = (MovieActor) o;
        return Objects.equals(movieName, that.movieName) && Objects.equals(actorName,
 that.actorName) && Objects.equals(actingAward, that.actingAward) &&
 Objects.equals(actingYear, that.actingYear) && Objects.equals(actingSchoolName,
 that.actingSchoolName);
    }

    @Override
```

```
    public int hashCode() {
        return Objects.hash(movieName, actorName, actingAward, actingYear,
 actingSchoolName);
    }

    @Override
    public int compareTo(MovieActor o) {
        if (this.movieName.compareTo(o.movieName) != 0){
            return this.movieName.compareTo(o.movieName);
        } else {
            return this.actorName.compareTo(o.actorName);
        }
    }
}
```

The code examples that follow query against the following items.

Items in the `MovieActor` table

```
MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',
 actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
 actingYear=2001, actingSchoolName='actingschool1'}
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
 actingYear=2001, actingSchoolName='actingschool2'}
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
 actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
 actingYear=2001, actingSchoolName='actingschool4'}
MovieActor{movieName='movie02', actorName='actor0', actingAward='actingaward0',
 actingYear=2002, actingSchoolName='null'}
MovieActor{movieName='movie02', actorName='actor1', actingAward='actingaward1',
 actingYear=2002, actingSchoolName='actingschool1'}
MovieActor{movieName='movie02', actorName='actor2', actingAward='actingaward2',
 actingYear=2002, actingSchoolName='actingschool2'}
MovieActor{movieName='movie02', actorName='actor3', actingAward='actingaward3',
 actingYear=2002, actingSchoolName='null'}
MovieActor{movieName='movie02', actorName='actor4', actingAward='actingaward4',
 actingYear=2002, actingSchoolName='actingschool4'}
MovieActor{movieName='movie03', actorName='actor0', actingAward='actingaward0',
 actingYear=2003, actingSchoolName='null'}
MovieActor{movieName='movie03', actorName='actor1', actingAward='actingaward1',
 actingYear=2003, actingSchoolName='actingschool1'}
```

```
MovieActor{movieName='movie03', actorName='actor2', actingAward='actingaward2',
 actingYear=2003, actingSchoolName='actingschool2'}
MovieActor{movieName='movie03', actorName='actor3', actingAward='actingaward3',
 actingYear=2003, actingSchoolName='null'}
MovieActor{movieName='movie03', actorName='actor4', actingAward='actingaward4',
 actingYear=2003, actingSchoolName='actingschool4'}
```

The following code defines two QueryConditional instances. `QueryConditionals` work with key values—either the partition key alone or in combination with the sort key—and correspond to the key conditional expressions of the DynamoDB service API. After comment line 1, the example defines the `keyEqual` instance that matches items with a partition value of **`movie01`**.

This example also defines a filter expression that filters off any item that has no **`actingschoolname`** on after comment line 2.

After comment line 3, the example shows the QueryEnhancedRequest instance that the code passes to the `DynamoDbTable.query()` method. This object combines the key condition and filter that the SDK uses to generate the request to the DynamoDB service.

```
    public static void query(DynamoDbTable movieActorTable) {

        // 1. Define a QueryConditional instance to return items matching a partition
 value.
        QueryConditional keyEqual = QueryConditional.keyEqualTo(b ->
 b.partitionValue("movie01"));
        // 1a. Define a QueryConditional that adds a sort key criteria to the partition
 value criteria.
        QueryConditional sortGreaterThanOrEqualTo =
 QueryConditional.sortGreaterThanOrEqualTo(b ->
 b.partitionValue("movie01").sortValue("actor2"));
        // 2. Define a filter expression that filters out items whose attribute value
 is null.
        final Expression filterOutNoActingschoolname =
 Expression.builder().expression("attribute_exists(actingschoolname)").build();

        // 3. Build the query request.
        QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
                .queryConditional(keyEqual)
                .filterExpression(filterOutNoActingschoolname)
                .build();
        // 4. Perform the query.
        PageIterable<MovieActor> pagedResults = movieActorTable.query(tableQuery);
```

```
        logger.info("page count: {}", pagedResults.stream().count()); // Log  number of
 pages.

        pagedResults.items().stream()
                .sorted()
                .forEach(
                        item -> logger.info(item.toString()) // Log the sorted list of
 items.
                );
```

The following is the output from running the method. The output displays items with a `movieName` value of **movie01** and displays no items with `actingSchoolName` equal to **null**.

```
2023-03-05 13:11:05 [main] INFO  org.example.tests.QueryDemo:46 - page count: 1
2023-03-05 13:11:05 [main] INFO  org.example.tests.QueryDemo:51 -
 MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
 actingYear=2001, actingSchoolName='actingschool1'}
2023-03-05 13:11:05 [main] INFO  org.example.tests.QueryDemo:51 -
 MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
 actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:11:05 [main] INFO  org.example.tests.QueryDemo:51 -
 MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
 actingYear=2001, actingSchoolName='actingschool4'}
```

In the following query request variation shown previously after comment line 3, the code replaces the `keyEqual` `QueryConditional` with the `sortGreaterThanOrEqualTo` `QueryConditional` that was defined after comment line 1a. The following code also removes the filter expression.

```
        QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
                .queryConditional(sortGreaterThanOrEqualTo)
```

Because this table has a composite primary key, all `QueryConditional` instances require a partition key value. `QueryConditional` methods that begin with `sort...` indicate that a *sort* key is required. The results are not sorted.

The following output displays the results from the query. The query returns items that have a `movieName` value equal to **movie01** and only items that have an `actorName` value that is greater than or equal to **actor2**. Because the filter was removed, the query returns items that have no value for the `actingSchoolName` attribute.

```
2023-03-05 13:15:00 [main] INFO  org.example.tests.QueryDemo:46 - page count: 1
```

```
2023-03-05 13:15:00 [main] INFO  org.example.tests.QueryDemo:51 -
 MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
 actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:15:00 [main] INFO  org.example.tests.QueryDemo:51 -
 MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
 actingYear=2001, actingSchoolName='null'}
2023-03-05 13:15:00 [main] INFO  org.example.tests.QueryDemo:51 -
 MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
 actingYear=2001, actingSchoolName='actingschool4'}
```

Perform batch operations

The DynamoDB Enhanced Client API offers two batch methods, `batchGetItem()` and `batchWriteItem()`.

`batchGetItem()` example

With the `DynamoDbEnhancedClient.batchGetItem()` method, you can retrieve up to 100 individual items across multiple tables in one overall request. The following example uses the `Customer` and `MovieActor` data classes shown previously.

In the example after lines 1 and 2, you build `ReadBatch` objects that you later add as parameters to the `batchGetItem()` method after comment line 3. The code after comment line 1 builds the batch to read from the `Customer` table. The code after comment line 1a shows the use of a `GetItemEnhancedRequest` builder that takes primary key values to specify the item to read. In contrast to specifying key values to request an item, you can use a data class to request an item as shown after comment line 1b. The SDK extracts the key values behind the scenes before submitting the request.

When you specify the item using the key-based approach as shown in the two statements after 2a, you can also specify that DynamoDB should perform a strongly consistent read. When the `consistentRead()` method is used, it must be used on all requested items for the same table.

To retrieve the items that DynamoDB found, use the `resultsForTable()` method that is shown after comment line 4. Call the method for each table that was read in the request. `resultsForTable()` returns a list of found items that you can process using any `java.util.List` method. This example logs each item.

To discover items that DynamoDB did not process, use the approach after comment line 5. The `BatchGetResultPage` class has the `unprocessedKeysForTable()` method that gives you access to each key that was unprocessed. The [BatchGetItem API reference](#) has more information about situations that result in unprocessed items.

```
    public static void batchGetItemExample(DynamoDbEnhancedClient enhancedClient,
                                           DynamoDbTable<Customer> customerTable,
                                           DynamoDbTable<MovieActor> movieActorTable) {

        Customer customer2 = new Customer();
        customer2.setId("2");
        customer2.setEmail("cust2@example.org");

        // 1. Build a batch to read from the Customer table.
        ReadBatch customerBatch = ReadBatch.builder(Customer.class)
                .mappedTableResource(customerTable)
                // 1a. Specify the primary key values for the item.
                .addGetItem(b -> b.key(k ->
 k.partitionValue("1").sortValue("cust1@orgname.org")))
                // 1b. Alternatively, supply a data class instances to provide the
 primary key values.
                .addGetItem(customer2)
                .build();

        // 2. Build a batch to read from the MovieActor table.
        ReadBatch moveActorBatch = ReadBatch.builder(MovieActor.class)
                .mappedTableResource(movieActorTable)
                // 2a. Call consistentRead(Boolean.TRUE) for each item for the same
 table.
                .addGetItem(b -> b.key(k ->
 k.partitionValue("movie01").sortValue("actor1")).consistentRead(Boolean.TRUE))
                .addGetItem(b -> b.key(k ->
 k.partitionValue("movie01").sortValue("actor4")).consistentRead(Boolean.TRUE))
                .build();

        // 3. Add ReadBatch objects to the request.
        BatchGetResultPageIterable resultPages = enhancedClient.batchGetItem(b ->
 b.readBatches(customerBatch, moveActorBatch));

        // 4. Retrieve the successfully requested items from each table.
        resultPages.resultsForTable(customerTable).forEach(item ->
 logger.info(item.toString()));
        resultPages.resultsForTable(movieActorTable).forEach(item ->
 logger.info(item.toString()));
```

```
        // 5. Retrieve the keys of the items requested but not processed by the
 service.
        resultPages.forEach((BatchGetResultPage pageResult) -> {
            pageResult.unprocessedKeysForTable(customerTable).forEach(key ->
 logger.info("Unprocessed item key: " + key.toString()));
            pageResult.unprocessedKeysForTable(customerTable).forEach(key ->
 logger.info("Unprocessed item key: " + key.toString()));
        });
    }
```

Assume that the following items are in the two tables before running the example code.

Items in tables

```
Customer [id=1, name=CustName1, email=cust1@example.org,
 regDate=2023-03-31T15:46:27.688Z]
Customer [id=2, name=CustName2, email=cust2@example.org,
 regDate=2023-03-31T15:46:28.688Z]
Customer [id=3, name=CustName3, email=cust3@example.org,
 regDate=2023-03-31T15:46:29.688Z]
Customer [id=4, name=CustName4, email=cust4@example.org,
 regDate=2023-03-31T15:46:30.688Z]
Customer [id=5, name=CustName5, email=cust5@example.org,
 regDate=2023-03-31T15:46:31.689Z]
MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',
 actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
 actingYear=2001, actingSchoolName='actingschool1'}
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
 actingYear=2001, actingSchoolName='actingschool2'}
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
 actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
 actingYear=2001, actingSchoolName='actingschool4'}
```

The following output shows the items returned and logged after comment line 4.

```
Customer [id=1, name=CustName1, email=cust1@example.org,
 regDate=2023-03-31T15:46:27.688Z]
Customer [id=2, name=CustName2, email=cust2@example.org,
 regDate=2023-03-31T15:46:28.688Z]
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
 actingYear=2001, actingSchoolName='actingschool4'}
```

```
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
 actingYear=2001, actingSchoolName='actingschool1'}
```

batchWriteItem() example

The batchWriteItem() method puts or deletes multiple items in one or more tables. You can specify up to 25 individual put or delete operations in the request. The following example uses the ProductCatalog and MovieActor model classes shown previously.

WriteBatch objects are built after comment lines 1 and 2. For the ProductCatalog table, the code puts one item and deletes one item. For the MovieActor table after comment line 2, the code puts two items and deletes one.

The batchWriteItem method is called after comment line 3. The builder parameter provides the batch requests for each table.

The returned BatchWriteResult object provides separate methods for each operation to view unprocessed requests. The code after comment line 4a provides the keys for unprocessed delete requests and the code after comment line 4b provides the unprocessed put items.

```
    public static void batchWriteItemExample(DynamoDbEnhancedClient enhancedClient,
                                             DynamoDbTable<ProductCatalog>
 catalogTable,
                                             DynamoDbTable<MovieActor> movieActorTable)
 {

        // 1. Build a batch to write to the ProductCatalog table.
        WriteBatch products = WriteBatch.builder(ProductCatalog.class)
                .mappedTableResource(catalogTable)
                .addPutItem(b -> b.item(getProductCatItem1()))
                .addDeleteItem(b -> b.key(k -> k
                        .partitionValue(getProductCatItem2().id())
                        .sortValue(getProductCatItem2().title())))
                .build();

        // 2. Build a batch to write to the MovieActor table.
        WriteBatch movies = WriteBatch.builder(MovieActor.class)
                .mappedTableResource(movieActorTable)
                .addPutItem(getMovieActorYeoh())
                .addPutItem(getMovieActorBlanchettPartial())
                .addDeleteItem(b -> b.key(k -> k
                        .partitionValue(getMovieActorStreep().getMovieName())
```

```
                        .sortValue(getMovieActorStreep().getActorName())))
                .build();

        // 3. Add WriteBatch objects to the request.
        BatchWriteResult batchWriteResult = enhancedClient.batchWriteItem(b ->
 b.writeBatches(products, movies));
        // 4. Retrieve keys for items the service did not process.
        // 4a. 'unprocessedDeleteItemsForTable()' returns keys for delete requests that
 did not process.
        if (batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).size() >
 0) {

 batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).forEach(key ->
                    logger.info(key.toString()));
        }
        // 4b. 'unprocessedPutItemsForTable()' returns keys for put requests that did
 not process.
        if (batchWriteResult.unprocessedPutItemsForTable(catalogTable).size() > 0) {
            batchWriteResult.unprocessedPutItemsForTable(catalogTable).forEach(key ->
                    logger.info(key.toString()));
        }
    }
```

The following helper methods provide the model objects for the put and delete operations.

Helper methods

```
    public static ProductCatalog getProductCatItem1() {
        return ProductCatalog.builder()
                .id(2)
                .isbn("1-565-85698")
                .authors(new HashSet<>(Arrays.asList("a", "b")))
                .price(BigDecimal.valueOf(30.22))
                .title("Title 55")
                .build();
    }

    public static ProductCatalog getProductCatItem2() {
        return ProductCatalog.builder()
                .id(4)
                .price(BigDecimal.valueOf(40.00))
                .title("Title 1")
                .build();
    }
```

```
    public static MovieActor getMovieActorBlanchettPartial() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Cate Blanchett");
        movieActor.setMovieName("Blue Jasmine");
        movieActor.setActingYear(2023);
        movieActor.setActingAward("Best Actress");
        return movieActor;
    }

    public static MovieActor getMovieActorStreep() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Meryl Streep");
        movieActor.setMovieName("Sophie's Choice");
        movieActor.setActingYear(1982);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("Yale School of Drama");
        return movieActor;
    }

    public static MovieActor getMovieActorYeoh(){
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Michelle Yeoh");
        movieActor.setMovieName("Everything Everywhere All at Once");
        movieActor.setActingYear(2023);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("Royal Academy of Dance");
        return movieActor;
    }
```

Assume that the tables contain the following items before you run the example code.

```
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best
 Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
 Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
```

After the example code finishes, the tables contain the following items.

```
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best
 Actress', actingYear=2013, actingSchoolName='null'}
```

```
MovieActor{movieName='Everything Everywhere All at Once', actorName='Michelle Yeoh',
 actingAward='Best Actress', actingYear=2023, actingSchoolName='Royal Academy of
 Dance'}
ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b], price=30.22}
```

Notice in the `MovieActor` table that the `Blue Jasmine` movie item has been replaced with the item used in the put request acquired through the `getMovieActorBlanchettPartial()` helper method. If a data bean attribute value was not provided, the value in the database is removed. This is why the resulting `actingSchoolName` is null for the `Blue Jasmine` movie item.

> **Note**
>
> Although the API documentation suggests that condition expressions can be used and that consumed capacity and collection metrics can be returned with individual put and delete requests, this is not the case in a batch write scenario. To improve performance for batch operations, these individual options are ignored.

Perform transaction operations

The DynamoDB Enhanced Client API provides the `transactGetItems()` and the `transactWriteItems()` methods. The transaction methods of the SDK for Java provide atomicity, consistency, isolation, and durability (ACID) in DynamoDB tables, helping you to maintain data correctness in your applications.

`transactGetItems()` example

The `transactGetItems()` method accepts up to 100 individual requests for items. All items are read in a single atomic transaction. The *Amazon DynamoDB Developer Guide* has information about the conditions that cause a `transactGetItems()` method to fail, and also about the isolation level used when you call `transactGetItem()`.

After comment line 1 in the following example, the code calls the `transactGetItems()` method with a `builder` parameter. The builder's `addGetItem()` is invoked three times with a data object that contains the key values that the SDK will use to generate the final request.

The request returns a list of `Document` objects after comment line 2. The list of documents that is returned contains non-null `Document` instances of item data in the same order as requested. The `Document.getItem(MappedTableResource<T> mappedTableResource)` method converts an untyped `Document` object into a typed Java object if item data was returned, otherwise the method returns null.

```
    public static void transactGetItemsExample(DynamoDbEnhancedClient enhancedClient,
                                                DynamoDbTable<ProductCatalog>
catalogTable,
                                                DynamoDbTable<MovieActor>
movieActorTable) {

        // 1. Request three items from two tables using a builder.
        final List<Document> documents = enhancedClient.transactGetItems(b -> b
                .addGetItem(catalogTable,
Key.builder().partitionValue(2).sortValue("Title 55").build())
                .addGetItem(movieActorTable, Key.builder().partitionValue("Sophie's
Choice").sortValue("Meryl Streep").build())
                .addGetItem(movieActorTable, Key.builder().partitionValue("Blue
Jasmine").sortValue("Cate Blanchett").build())
                .build());

        // 2. A list of Document objects is returned in the same order as requested.
        ProductCatalog title55 = documents.get(0).getItem(catalogTable);
        if (title55 != null) {
            logger.info(title55.toString());
        }

        MovieActor sophiesChoice = documents.get(1).getItem(movieActorTable);
        if (sophiesChoice != null) {
            logger.info(sophiesChoice.toString());
        }

        // 3. The getItem() method returns null if the Document object contains no item
from DynamoDB.
        MovieActor blueJasmine = documents.get(2).getItem(movieActorTable);
        if (blueJasmine != null) {
            logger.info(blueJasmine.toString());
        }
    }
```

The DynamoDB tables contain the following items before the code example runs.

```
ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}
```

```
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
 Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
```

The following output is logged. If an item is requested but not found, it not returned as is the case for the request for the movie named `Blue Jasmine`.

```
ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
 Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
```

`transactWriteItems()` examples

The `transactWriteItems()` accepts up to 100 put, update, or delete actions in a single atomic transaction across multiple tables. The *Amazon DynamoDB Developer Guide* contains details about restrictions and failure conditions of the underlying DynamoDB service operation.

Basic example

In the following example, four operations are requested for two tables. The corresponding model classes `ProductCatalog` and `MovieActor` were shown previously.

Each of the three possible operations—put, update, and delete—uses a dedicated request parameter to specify the details.

The code after comment line 1 shows the simple variation of the `addPutItem()` method. The method accepts a `MappedTableResource` object and the data object instance to put. The statement after comment line 2 shows the variation that accepts a `TransactPutItemEnhancedRequest` instance. This variation lets you add more options in the request, such as a condition expression. A subsequent example shows a condition expression for an individual operation.

An update operation is requested after comment line 3. `TransactUpdateItemEnhancedRequest` has an `ignoreNulls()` method that lets you configure what the SDK does with `null` values on the model object. If the `ignoreNulls()` method returns true, the SDK does not remove the table's attribute values for data object attributes that are `null`. If the `ignoreNulls()` method returns false, the SDK requests the DynamoDB service to remove the attributes from the item in the table. The default value for `ignoreNulls` is false.

The statement after comment line 4 shows the variation of a delete request that takes a data object. The enhanced client extracts the key values before dispatching the final request.

```
    public static void transactWriteItems(DynamoDbEnhancedClient enhancedClient,
                                          DynamoDbTable<ProductCatalog> catalogTable,
                                          DynamoDbTable<MovieActor> movieActorTable) {

        enhancedClient.transactWriteItems(b -> b
                // 1. Simplest variation of put item request.
                .addPutItem(catalogTable, getProductCatId2())
                // 2. Put item request variation that accommodates condition
 expressions.
                .addPutItem(movieActorTable,
 TransactPutItemEnhancedRequest.builder(MovieActor.class)
                        .item(getMovieActorStreep())

 .conditionExpression(Expression.builder().expression("attribute_not_exists
 (movie)").build())
                        .build())
                // 3. Update request that does not remove attribute values on the table
 if the data object's value is null.
                .addUpdateItem(catalogTable,
 TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
                        .item(getProductCatId4ForUpdate())
                        .ignoreNulls(Boolean.TRUE)
                        .build())
                // 4. Variation of delete request that accepts a data object. The key
 values are extracted for the request.
                .addDeleteItem(movieActorTable, getMovieActorBlanchett())
        );
    }
```

The following helper methods provide the data objects for the `add*Item` parameters.

Helper methods

```
    public static ProductCatalog getProductCatId2() {
        return ProductCatalog.builder()
                .id(2)
                .isbn("1-565-85698")
                .authors(new HashSet<>(Arrays.asList("a", "b")))
                .price(BigDecimal.valueOf(30.22))
                .title("Title 55")
                .build();
    }
```

```
    public static ProductCatalog getProductCatId4ForUpdate() {
        return ProductCatalog.builder()
                .id(4)
                .price(BigDecimal.valueOf(40.00))
                .title("Title 1")
                .build();
    }

    public static MovieActor getMovieActorBlanchett() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Cate Blanchett");
        movieActor.setMovieName("Tar");
        movieActor.setActingYear(2022);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("National Institute of Dramatic Art");
        return movieActor;
    }

    public static MovieActor getMovieActorStreep() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Meryl Streep");
        movieActor.setMovieName("Sophie's Choice");
        movieActor.setActingYear(1982);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("Yale School of Drama");
        return movieActor;
    }
```

The DynamoDB tables contain the following items before the code example runs.

```
1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress',
 actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}
```

The following items are in the tables after the code finishes running.

```
3 | ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b],
 price=30.22}
4 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=40.0}
5 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
 Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
```

The item on line 2 has been deleted and lines 3 and 5 show the items that were put. Line 4 shows the update of line 1. The `price` value is the only value that changed on the item. If `ignoreNulls()` had returned false, line 4 would look like the following line.

```
ProductCatalog{id=4, title='Title 1', isbn='null', authors=null, price=40.0}
```

Condition check example

The following example shows the use of a condition check. A condition check is used to check that an item exists or to check the condition of specific attributes of an item in the database. The item checked in the condition check cannot be used in another operation in the transaction.

> **Note**
>
> You can't target the same item with multiple operations within the same transaction. For example, you can't perform a condition check and also attempt to update the same item in the same transaction.

The example shows one of each type of operation in a transactional write items request. After comment line 2, the `addConditionCheck()` method supplies the condition that fails the transaction if the `conditionExpression` parameter evaluates to `false`. The condition expression that is returned from the method shown in the Helper methods block checks if the award year for the movie `Sophie's Choice` is not equal to `1982`. If it is, the expression evaluates to `false` and the transaction fails.

This guide discusses [expressions](https://docs.aws.amazon.com/) in depth in another topic.

```
    public static void conditionCheckFailExample(DynamoDbEnhancedClient enhancedClient,
                                                 DynamoDbTable<ProductCatalog>
 catalogTable,
                                                 DynamoDbTable<MovieActor>
 movieActorTable) {

        try {
            enhancedClient.transactWriteItems(b -> b
                    // 1. Perform one of each type of operation with the next three
 methods.
                    .addPutItem(catalogTable,
 TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
```

```
                            .item(getProductCatId2()).build())
                    .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
                            .item(getProductCatId4ForUpdate())
                            .ignoreNulls(Boolean.TRUE).build())
                    .addDeleteItem(movieActorTable,
TransactDeleteItemEnhancedRequest.builder()
                            .key(b1 -> b1

.partitionValue(getMovieActorBlanchett().getMovieName())

.sortValue(getMovieActorBlanchett().getActorName())).build())
                    // 2. Add a condition check on a table item that is not involved in
another operation in this request.
                    .addConditionCheck(movieActorTable, ConditionCheck.builder()
                            .conditionExpression(buildConditionCheckExpression())
                            .key(k -> k
                                    .partitionValue("Sophie's Choice")
                                    .sortValue("Meryl Streep"))
                            // 3. Specify the request to return existing values from
the item if the condition evaluates to true.

.returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
                            .build())
                    .build());
        // 4. Catch the exception if the transaction fails and log the information.
        } catch (TransactionCanceledException ex) {
            ex.cancellationReasons().stream().forEach(cancellationReason -> {
                logger.info(cancellationReason.toString());
            });
        }
    }
```

The following helper methods are used in the previous code example.

Helper methods

```
    private static Expression buildConditionCheckExpression() {
        Map<String, AttributeValue> expressionValue = Map.of(
                ":year", numberValue(1982));

        return Expression.builder()
                .expression("actingyear <> :year")
                .expressionValues(expressionValue)
```

```
                .build();
    }

    public static ProductCatalog getProductCatId2() {
        return ProductCatalog.builder()
                .id(2)
                .isbn("1-565-85698")
                .authors(new HashSet<>(Arrays.asList("a", "b")))
                .price(BigDecimal.valueOf(30.22))
                .title("Title 55")
                .build();
    }

    public static ProductCatalog getProductCatId4ForUpdate() {
        return ProductCatalog.builder()
                .id(4)
                .price(BigDecimal.valueOf(40.00))
                .title("Title 1")
                .build();
    }

    public static MovieActor getMovieActorBlanchett() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Cate Blanchett");
        movieActor.setMovieName("Blue Jasmine");
        movieActor.setActingYear(2013);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("National Institute of Dramatic Art");
        return movieActor;
    }
```

The DynamoDB tables contain the following items before the code example runs.

```
1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
 Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
3 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress',
 actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}
```

The following items are in the tables after the code finishes running.

```
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
```

```
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
 Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress',
 actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}
```

Items remain unchanged in the tables because the transaction failed. The `actingYear` value for the movie `Sophie's Choice` is 1982, as shown on line 2 of the items in the table before the `transactWriteItem()` method is called.

To capture the cancellation information for the transaction, enclose the `transactWriteItems()` method call in a `try` block and `catch` the [`TransactionCanceledException`](). After comment line 4 of the example, the code logs each [`CancellationReason`]() object. Because the code following comment line 3 of the example specifies that values should be returned for the item that caused the transaction to fail, the log displays the raw database values for the `Sophie's Choice` movie item.

```
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Meryl Streep),
 movie=AttributeValue(S=Sophie's Choice), actingaward=AttributeValue(S=Best Actress),
 actingyear=AttributeValue(N=1982), actingschoolname=AttributeValue(S=Yale School of
 Drama)}, ¬
    Code=ConditionalCheckFailed, Message=The conditional request failed.)
```

Single operation condition example

The following example shows the use of a condition on a single operation in a transaction request. The delete operation after comment line 1 contains a condition that checks the value of the target item of the operation against the database. In this example, the condition expression created with the helper method after comment line 2 specifies that the item should be deleted from the database if the acting year of the movie is not equal to 2013.

[Expressions]() are discussed later in this guide.

```
    public static void singleOperationConditionFailExample(DynamoDbEnhancedClient
 enhancedClient,

 DynamoDbTable<ProductCatalog> catalogTable,
                                                           DynamoDbTable<MovieActor>
 movieActorTable) {
```

```
        try {
            enhancedClient.transactWriteItems(b -> b
                    .addPutItem(catalogTable,
TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
                            .item(getProductCatId2())
                            .build())
                    .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
                            .item(getProductCatId4ForUpdate())
                            .ignoreNulls(Boolean.TRUE).build())
                    // 1. Delete operation that contains a condition expression
                    .addDeleteItem(movieActorTable,
TransactDeleteItemEnhancedRequest.builder()
                            .key((Key.Builder k) -> {
                                MovieActor blanchett = getMovieActorBlanchett();
                                k.partitionValue(blanchett.getMovieName())
                                        .sortValue(blanchett.getActorName());
                            })
                            .conditionExpression(buildDeleteItemExpression())

.returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
                            .build())
                    .build());
        } catch (TransactionCanceledException ex) {
            ex.cancellationReasons().forEach(cancellationReason ->
logger.info(cancellationReason.toString()));
        }
    }

    // 2. Provide condition expression to check if 'actingyear' is not equal to 2013.
    private static Expression buildDeleteItemExpression() {
        Map<String, AttributeValue> expressionValue = Map.of(
                ":year", numberValue(2013));

        return Expression.builder()
                .expression("actingyear <> :year")
                .expressionValues(expressionValue)
                .build();
    }
```

The following helper methods are used in the previous code example.

Helper methods

```
    public static ProductCatalog getProductCatId2() {
        return ProductCatalog.builder()
                .id(2)
                .isbn("1-565-85698")
                .authors(new HashSet<>(Arrays.asList("a", "b")))
                .price(BigDecimal.valueOf(30.22))
                .title("Title 55")
                .build();
    }

    public static ProductCatalog getProductCatId4ForUpdate() {
        return ProductCatalog.builder()
                .id(4)
                .price(BigDecimal.valueOf(40.00))
                .title("Title 1")
                .build();
    }
    public static MovieActor getMovieActorBlanchett() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Cate Blanchett");
        movieActor.setMovieName("Blue Jasmine");
        movieActor.setActingYear(2013);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("National Institute of Dramatic Art");
        return movieActor;
    }
```

The DynamoDB tables contain the following items before the code example runs.

```
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
```

The following items are in the tables after the code finishes running.

```
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2023-03-15 11:29:07 [main] INFO  org.example.tests.TransactDemoTest:168 - MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
```

Items remain unchanged in the tables because the transaction failed. The `actingYear` value for the movie `Blue Jasmine` is `2013` as shown on line 2 in the list of items before the code example runs.

The following lines are logged to the console.

```
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Cate Blanchett),
 movie=AttributeValue(S=Blue Jasmine), actingaward=AttributeValue(S=Best Actress),
 actingyear=AttributeValue(N=2013), actingschoolname=AttributeValue(S=National
 Institute of Dramatic Art)},
    Code=ConditionalCheckFailed, Message=The conditional request failed)
```

Use secondary indices

Secondary indices improve data access by defining alternative keys that you use in query and scan operations. Global secondary indices (GSI) have a partition key and a sort key that can be different from those on the base table. In contrast, local secondary indices (LSI) use the partition key of the primary index.

Annotate data class with secondary index annotations

Attributes that participate in secondary indices require either the `@DynamoDbSecondaryPartitionKey` or `@DynamoDbSecondarySortKey` annotation.

The following class shows annotations for two indices. The GSI named *SubjectLastPostedDateIndex* uses the `Subject` attribute for the partition key and the `LastPostedDateTime` for the sort key. The LSI named *ForumLastPostedDateIndex* uses the `ForumName` as its partition key and `LastPostedDateTime` as its sort key.

Note that the `Subject` attribute serves a dual role. It is the primary key's sort key and the partition key of the GSI named *SubjectLastPostedDateIndex*.

`MessageThread` class

The `MessageThread` class is suitable to use as a data class for the [example Thread table](#) in the *Amazon DynamoDB Developer Guide*.

Imports

```
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
```

```
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
 software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.util.List;
```

```
@DynamoDbBean
public class MessageThread {
    private String ForumName;
    private String Subject;
    private String Message;
    private String LastPostedBy;
    private String LastPostedDateTime;
    private Integer Views;
    private Integer Replies;
    private Integer Answered;
    private List<String> Tags;

    @DynamoDbPartitionKey
    public String getForumName() {
        return ForumName;
    }

    public void setForumName(String forumName) {
        ForumName = forumName;
    }

    // Sort key for primary index and partition key for GSI
 "SubjectLastPostedDateIndex".
    @DynamoDbSortKey
    @DynamoDbSecondaryPartitionKey(indexNames = "SubjectLastPostedDateIndex")
    public String getSubject() {
        return Subject;
    }

    public void setSubject(String subject) {
        Subject = subject;
    }
```

```
    // Sort key for GSI "SubjectLastPostedDateIndex" and sort key for LSI
 "ForumLastPostedDateIndex".
    @DynamoDbSecondarySortKey(indexNames = {"SubjectLastPostedDateIndex",
 "ForumLastPostedDateIndex"})
    public String getLastPostedDateTime() {
        return LastPostedDateTime;
    }

    public void setLastPostedDateTime(String lastPostedDateTime) {
        LastPostedDateTime = lastPostedDateTime;
    }
    public String getMessage() {
        return Message;
    }

    public void setMessage(String message) {
        Message = message;
    }

    public String getLastPostedBy() {
        return LastPostedBy;
    }

    public void setLastPostedBy(String lastPostedBy) {
        LastPostedBy = lastPostedBy;
    }

    public Integer getViews() {
        return Views;
    }

    public void setViews(Integer views) {
        Views = views;
    }

    @DynamoDbSecondaryPartitionKey(indexNames = "ForumRepliesIndex")
    public Integer getReplies() {
        return Replies;
    }

    public void setReplies(Integer replies) {
        Replies = replies;
    }
```

```
    public Integer getAnswered() {
        return Answered;
    }

    public void setAnswered(Integer answered) {
        Answered = answered;
    }

    public List<String> getTags() {
        return Tags;
    }

    public void setTags(List<String> tags) {
        Tags = tags;
    }

    public MessageThread() {
        this.Answered = 0;
        this.LastPostedBy = "";
        this.ForumName = "";
        this.Message = "";
        this.LastPostedDateTime = "";
        this.Replies = 0;
        this.Views = 0;
        this.Subject = "";
    }

    @Override
    public String toString() {
        return "MessageThread{" +
                "ForumName='" + ForumName + '\'' +
                ", Subject='" + Subject + '\'' +
                ", Message='" + Message + '\'' +
                ", LastPostedBy='" + LastPostedBy + '\'' +
                ", LastPostedDateTime='" + LastPostedDateTime + '\'' +
                ", Views=" + Views +
                ", Replies=" + Replies +
                ", Answered=" + Answered +
                ", Tags=" + Tags +
                '}';
    }
}
```

Create the index

Beginning with version 2.20.86 of the SDK for Java, the `createTable()` method automatically generates secondary indexes from data class annotations. By default, all attributes from the base table are copied to an index and the provisioned throughput values are 20 read capacity units and 20 write capacity units.

However, if you use an SDK version prior to 2.20.86, you need to build the index along with the table as shown in the following example. This example builds the two indexes for the `Thread` table. The builder parameter has methods to configure both types of indexes as shown after comment lines 1 and 2. You use the index builder's `indexName()` method to associate the index names specified in the data class annotations with the intended type of index.

This code configures all of the table attributes to end up in both indexes after comment lines 3 and 4. More information about attribute projections is available in the *Amazon DynamoDB Developer Guide*.

```
   public static void createMessageThreadTable(DynamoDbTable<MessageThread>
messageThreadDynamoDbTable, DynamoDbClient dynamoDbClient) {
       messageThreadDynamoDbTable.createTable(b -> b
               // 1. Generate the GSI.
               .globalSecondaryIndices(gsi ->
gsi.indexName("SubjectLastPostedDateIndex")
                       // 3. Populate the GSI with all attributes.
                       .projection(p -> p
                               .projectionType(ProjectionType.ALL))
               )
               // 2. Generate the LSI.
               .localSecondaryIndices(lsi -> lsi.indexName("ForumLastPostedDateIndex")
                       // 4. Populate the LSI with all attributes.
                       .projection(p -> p
                               .projectionType(ProjectionType.ALL))
               )
       );
```

Query by using an index

The following example queries the local secondary index *ForumLastPostedDateIndex*.

Following comment line 2, you create a QueryConditional object that is required when calling the DynamoDbIndex.query() method.

You get a reference to the index you want to query after comment line 3 by passing in the name of the index. Following comment line 4, you call the `query()` method on the index passing in the `QueryConditional` object.

You also configure the query to return three attribute values as shown after comment line 5. If `attributesToProject()` is not called, the query returns all attribute values. Notice that the specified attribute names begin with lowercase letters. These attribute names match those used in the table, not necessarily the attribute names of the data class.

Following comment line 6, iterate through the results and log each item returned by the query and also store it in the list to return to the caller.

```
public static List<MessageThread> queryUsingSecondaryIndices(DynamoDbEnhancedClient
 enhancedClient,
                                                                String lastPostedDate,

 DynamoDbTable<MessageThread> threadTable) {
        // 1. Log the parameter value.
        logger.info("lastPostedDate value: {}", lastPostedDate);

        // 2. Create a QueryConditional whose sort key value must be greater than or
 equal to the parameter value.
        QueryConditional queryConditional =
 QueryConditional.sortGreaterThanOrEqualTo(qc ->
                qc.partitionValue("Forum02").sortValue(lastPostedDate));

        // 3. Specify the index name to query the DynamoDbIndex instance.
        final DynamoDbIndex<MessageThread> forumLastPostedDateIndex =
 threadTable.index("ForumLastPostedDateIndex");

        // 4. Perform the query by using the QueryConditional object.
        final SdkIterable<Page<MessageThread>> pagedResult =
 forumLastPostedDateIndex.query(q -> q
                .queryConditional(queryConditional)
                // 5. Request three attribute in the results.
                .attributesToProject("forumName", "subject", "lastPostedDateTime"));

        List<MessageThread> collectedItems = new ArrayList<>();
        // 6. Iterate through the pages response and sort the items.
        pagedResult.stream().forEach(page -> page.items().stream()

 .sorted(Comparator.comparing(MessageThread::getLastPostedDateTime))
                        .forEach(mt -> {
```

```
                            // 7. Log the returned items and add the collection to
 return to the caller.
                            logger.info(mt.toString());
                            collectedItems.add(mt);
                        }));
        return collectedItems;
    }
```

The following items exist in the database before the query is run.

```
MessageThread{ForumName='Forum01', Subject='Subject01', Message='Message01',
 LastPostedBy='', LastPostedDateTime='2023.03.28', Views=0, Replies=0, Answered=0,
 Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject02', Message='Message02',
 LastPostedBy='', LastPostedDateTime='2023.03.29', Views=0, Replies=0, Answered=0,
 Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject04', Message='Message04',
 LastPostedBy='', LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0,
 Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject08', Message='Message08',
 LastPostedBy='', LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0,
 Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject10', Message='Message10',
 LastPostedBy='', LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0,
 Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject03', Message='Message03',
 LastPostedBy='', LastPostedDateTime='2023.03.30', Views=0, Replies=0, Answered=0,
 Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject06', Message='Message06',
 LastPostedBy='', LastPostedDateTime='2023.04.02', Views=0, Replies=0, Answered=0,
 Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject09', Message='Message09',
 LastPostedBy='', LastPostedDateTime='2023.04.05', Views=0, Replies=0, Answered=0,
 Tags=null}
MessageThread{ForumName='Forum05', Subject='Subject05', Message='Message05',
 LastPostedBy='', LastPostedDateTime='2023.04.01', Views=0, Replies=0, Answered=0,
 Tags=null}
MessageThread{ForumName='Forum07', Subject='Subject07', Message='Message07',
 LastPostedBy='', LastPostedDateTime='2023.04.03', Views=0, Replies=0, Answered=0,
 Tags=null}
```

The logging statements at lines 1 and 6 result in the following console output.

```
lastPostedDate value: 2023.03.31
MessageThread{ForumName='Forum02', Subject='Subject04', Message='', LastPostedBy='',
 LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0, Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject08', Message='', LastPostedBy='',
 LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0, Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject10', Message='', LastPostedBy='',
 LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0, Tags=null}
```

The query returned items with a `forumName` value of *Forum02* and a `lastPostedDateTime` value greater than or equal to *2023.03.31*. The results show `message` values with an empty string although the `message` attributes have values in the index. This is because the message attribute was not projected after comment line 5.

Use advanced mapping features

Learn about advanced table schema features in the DynamoDB Enhanced Client API.

Understand table schema types

[`TableSchema`] is the interface to the mapping functionality of the DynamoDB Enhanced Client API. It can map a data object to and from a map of [AttributeValues]. A `TableSchema` object needs to know about the structure of the table it is mapping. This structure information is stored in a [`TableMetadata`] object.

The enhanced client API has several implementations of `TableSchema`, which follow.

Table schema generated from annotated classes

It is a moderately expensive operation to build a `TableSchema` from annotated classes, so we recommend doing this once, at application startup.

[BeanTableSchema]

This implementation is built based on attributes and annotations of a bean class. An example of this approach is demonstrated in the [Get started section].

> **Note**
>
> If a `BeanTableSchema` is not behaving as you expect, enable debug logging for `software.amazon.awssdk.enhanced.dynamodb.beans`.

ImmutableTableSchema

This implementation is built from an immutable data class. This approach is described in the ??? section.

Table schema generated with a builder

The following `TableSchemas` are built from code by using a builder. This approach is less costly than the approach that uses annotated data classes. The builder approach avoids the use of annotations and doesn't require JavaBean naming standards.

StaticTableSchema

This implementation is built for mutable data classes. The getting started section of this guide demonstrated how to generate a `StaticTableSchema` using a builder.

StaticImmutableTableSchema

Similarly to how you build a `StaticTableSchema`, you generate an implementation of this type of `TableSchema` using a builder for use with immutable data classes.

Table schema for data without a fixed schema

DocumentTableSchema

Unlike other implementations of `TableSchema`, you don't define attributes for a `DocumentTableSchema` instance. Usually, you specify only primary keys and attribute converter providers. An `EnhancedDocument` instance provides the attributes that you build from individual elements or from a JSON string.

Explicitly include or exclude attributes

The DynamoDB Enhanced Client API offers annotations to exclude data class attributes from becoming attributes on a table. With the API, you can also use an attribute name that's different from the data class attribute name.

Exclude attributes

To ignore attributes that should not be mapped to a DynamoDB table, mark the attribute with the `@DynamoDbIgnore` annotation.

```
private String internalKey;

@DynamoDbIgnore
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { this.internalKey = internalKey;}
```

Include attributes

To change the name of an attribute used in the DynamoDB table, mark it with the `@DynamoDbAttribute` annotation and supply a different name.

```
private String internalKey;

@DynamoDbAttribute("renamedInternalKey")
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { this.internalKey = internalKey;}
```

Control attribute conversion

By default, a table schema provides converters for many common Java types through a default implementation of the [`AttributeConverterProvider`] interface. You can change the overall default behavior with a custom `AttributeConverterProvider` implementation. You can also change the converter for a single attribute.

For a list of available converters, see the [AttributeConverter] interface Java doc.

Provide custom attribute converter providers

You can provide a single `AttributeConverterProvider` or a chain of ordered `AttributeConverterProviders` through the `@DynamoDbBean` (`converterProviders = {…}`) annotation. Any custom `AttributeConverterProvider` must extend the `AttributeConverterProvider` interface.

Note that if you supply your own chain of attribute converter providers, you will override the default converter provider, `DefaultAttributeConverterProvider`. If you want to use the functionality of the `DefaultAttributeConverterProvider`, you must include it in the chain.

It's also possible to annotate the bean with an empty array `{}`. This disables the use of any attribute converter providers, including the default. In this case all attributes that are to be mapped must have their own attribute converter.

The following snippet shows a single converter provider.

```
@DynamoDbBean(converterProviders = ConverterProvider1.class)
public class Customer {

}
```

The following snippet shows the use of a chain of converter providers. Since the SDK default is provided last, it has the lowest priority.

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

}
```

The static table schema builders have an `attributeConverterProviders()` method that works the same way. This is shown in the following snippet.

```
private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
  StaticTableSchema.builder(Customer.class)
    .newItemSupplier(Customer::new)
    .addAttribute(String.class, a -> a.name("name")
                                     a.getter(Customer::getName)
                                     a.setter(Customer::setName))
    .attributeConverterProviders(converterProvider1, converterProvider2)
    .build();
```

Override the mapping of a single attribute

To override the way a single attribute is mapped, supply an `AttributeConverter` for the attribute. This addition overrides any converters provided by `AttributeConverterProviders` in the table schema. This adds a custom converter for only that attribute. Other attributes, even those of the same type, won't use that converter unless it is explicitly specified for those other attributes.

The `@DynamoDbConvertedBy` annotation is used to specify the custom `AttributeConverter` class as shown in the following snippet.

```
@DynamoDbBean
public class Customer {
    private String name;

    @DynamoDbConvertedBy(CustomAttributeConverter.class)
    public String getName() { return this.name; }
    public void setName(String name) { this.name = name;}
}
```

The builders for static schemas have an equivalent attribute builder `attributeConverter()` method. This method takes an instance of an `AttributeConverter` as the following shows.

```
private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
  StaticTableSchema.builder(Customer.class)
    .newItemSupplier(Customer::new)
    .addAttribute(String.class, a -> a.name("name")
                                     a.getter(Customer::getName)
                                     a.setter(Customer::setName)
                                     a.attributeConverter(customAttributeConverter))
    .build();
```

Example

This example shows an `AttributeConverterProvider` implementation that provides an attribute converter for [`java.net.HttpCookie`](https://docs.oracle.com/javase/8/docs/api/java/net/HttpCookie.html) objects.

The following `SimpleUser` class contains an attribute named `lastUsedCookie` that is an instance of `HttpCookie`.

The parameter to the `@DynamoDbBean` annotations lists the two `AttributeConverterProvider` classes that provide converters.

Class with annotations

```
   @DynamoDbBean(converterProviders = {CookieConverterProvider.class,
DefaultAttributeConverterProvider.class})
   public static final class SimpleUser {
       private String name;
       private HttpCookie lastUsedCookie;

       @DynamoDbPartitionKey
```

```
        public String getName() {
            return name;
        }

        public void setName(String name) {
            this.name = name;
        }

        public HttpCookie getLastUsedCookie() {
            return lastUsedCookie;
        }

        public void setLastUsedCookie(HttpCookie lastUsedCookie) {
            this.lastUsedCookie = lastUsedCookie;
        }
```

Static table schema

```
    private static final TableSchema<SimpleUser> SIMPLE_USER_TABLE_SCHEMA =
            TableSchema.builder(SimpleUser.class)
                    .newItemSupplier(SimpleUser::new)
                    .attributeConverterProviders(CookieConverterProvider.create(),
 AttributeConverterProvider.defaultProvider())
                    .addAttribute(String.class, a -> a.name("name")
                            .setter(SimpleUser::setName)
                            .getter(SimpleUser::getName)
                            .tags(StaticAttributeTags.primaryPartitionKey()))
                    .addAttribute(HttpCookie.class, a -> a.name("lastUsedCookie")
                            .setter(SimpleUser::setLastUsedCookie)
                            .getter(SimpleUser::getLastUsedCookie))
                    .build();
```

The `CookieConverterProvider` in the following example provides an instance of an `HttpCookeConverter`.

```
    public static final class CookieConverterProvider implements
 AttributeConverterProvider {
        private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
 ImmutableMap.of(
                // 1. Add HttpCookieConverter to the internal cache.
                EnhancedType.of(HttpCookie.class), new HttpCookieConverter());
```

```
        public static CookieConverterProvider create() {
            return new CookieConverterProvider();
        }

        // The SDK calls this method to find out if the provider contains a
AttributeConverter instance
        // for the EnhancedType<T> argument.
        @SuppressWarnings("unchecked")
        @Override
        public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
            return (AttributeConverter<T>) converterCache.get(enhancedType);
        }
    }
```

Conversion code

In the `transformFrom()` method of the following `HttpCookieConverter` class, the code receives an `HttpCookie` instance and transforms it into a DynamoDB map that is stored as an attribute.

The `transformTo()` method receives a DynamoDB map parameter, then invokes the `HttpCookie` constructor that requires a name and a value.

```
    public static final class HttpCookieConverter implements
AttributeConverter<HttpCookie> {

        @Override
        public AttributeValue transformFrom(HttpCookie httpCookie) {

            return AttributeValue.fromM(
            Map.of ("cookieName", AttributeValue.fromS(httpCookie.getName()),
                    "cookieValue", AttributeValue.fromS(httpCookie.getValue()))
            );
        }

        @Override
        public HttpCookie transformTo(AttributeValue attributeValue) {
            Map<String, AttributeValue> map = attributeValue.m();
            return new HttpCookie(
                    map.get("cookieName").s(),
                    map.get("cookieValue").s());
        }
```

```
        @Override
        public EnhancedType<HttpCookie> type() {
            return EnhancedType.of(HttpCookie.class);
        }

        @Override
        public AttributeValueType attributeValueType() {
            return AttributeValueType.M;
        }
    }
```

Change update behavior of attributes

You can customize the update behavior of individual attributes when you perform an *update* operation. Some examples of update operations in the DynamoDB Enhanced Client API are updateItem() and transactWriteItems().

For example, imagine that you want to store a *created on* timestamp on your record. However, you want its value to be written only if there's no existing value for the attribute already in the database. In this case, you use the `WRITE_IF_NOT_EXISTS` update behavior.

The following example shows the annotation that adds the behavior to the `createdOn` attribute.

```
@DynamoDbBean
public class Customer extends GenericRecord {
    private String id;
    private Instant createdOn;

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

    @DynamoDbUpdateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)
    public Instant getCreatedOn() { return this.createdOn; }
    public void setCreatedOn(Instant createdOn) { this.createdOn = createdOn; }
}
```

You can declare the same update behavior when you build a static table schema as shown in the following example after comment line 1.

```
static final TableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
     TableSchema.builder(Customer.class)
```

```
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("id")
                                          .getter(Customer::getId)
                                          .setter(Customer::setId)

  .tags(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(Instant.class, a -> a.name("createdOn")
                                           .getter(Customer::getCreatedOn)
                                           .setter(Customer::setCreatedOn)
                                           // 1. Add an UpdateBehavior.

  .tags(StaticAttributeTags.updateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)))
        .build();
```

Flatten attributes from other classes

If the attributes for your table are spread across several different Java classes, either through inheritance or composition, the DynamoDB Enhanced Client API provides support to flatten the attributes into one class.

Use inheritance

If your classes use inheritance, use the following approaches to flatten the hierarchy.

Use annotated beans

For the annotation approach, both classes must carry the `@DynamoDbBean` annotation and a class must carry one or more primary key annotations.

The following shows examples of data classes that have an inheritance relationship.

Standard data class

```
@DynamoDbBean
public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

@DynamoDbBean
public abstract class GenericRecord {
```

```
    private String id;
    private String createdDate;

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

    public String getCreatedDate() { return createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
}
```

Lombok

Lombok's [onMethod option](https://projectlombok.org/features/experimental/onX) copies attribute-based DynamoDB annotations, such as `@DynamoDbPartitionKey`, onto the generated code.

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

@Data
@DynamoDbBean
public abstract class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdDate;
}
```

Use static schemas

For the static schema approach, use the `extend()` method of the builder to collapse the attributes of the parent class onto the child class. This is shown after comment line 1 in the following example.

```
        StaticTableSchema<org.example.tests.model.inheritance.stat.GenericRecord>
 GENERIC_RECORD_SCHEMA =

 StaticTableSchema.builder(org.example.tests.model.inheritance.stat.GenericRecord.class)
```

```
                        // The partition key will be inherited by the top level mapper.
                        .addAttribute(String.class, a -> a.name("id")

.getter(org.example.tests.model.inheritance.stat.GenericRecord::getId)

.setter(org.example.tests.model.inheritance.stat.GenericRecord::setId)
                                .tags(primaryPartitionKey()))
                        .addAttribute(String.class, a -> a.name("created_date")

.getter(org.example.tests.model.inheritance.stat.GenericRecord::getCreatedDate)

.setter(org.example.tests.model.inheritance.stat.GenericRecord::setCreatedDate))
                        .build();

       StaticTableSchema<org.example.tests.model.inheritance.stat.Customer>
CUSTOMER_SCHEMA =

StaticTableSchema.builder(org.example.tests.model.inheritance.stat.Customer.class)

.newItemSupplier(org.example.tests.model.inheritance.stat.Customer::new)
                        .addAttribute(String.class, a -> a.name("name")

.getter(org.example.tests.model.inheritance.stat.Customer::getName)

.setter(org.example.tests.model.inheritance.stat.Customer::setName))
                        // 1. Use the extend() method to collapse the parent attributes
onto the child class.
                        .extend(GENERIC_RECORD_SCHEMA)     // All the attributes of the
GenericRecord schema are added to Customer.
                        .build();
```

The previous static schema example uses the following data classes. Because the mapping is defined when you build the static table schema, the data classes don't require annotations.

Data classes

Standard data class

```
public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
```

```
}


public abstract class GenericRecord {
    private String id;
    private String createdDate;

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

    public String getCreatedDate() { return createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
```

Lombok

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

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

Use composition

If your classes use composition, use the following approaches to flatten the hierarchy.

Use annotated beans

The `@DynamoDbFlatten` annotation flattens the contained class.

The following data class examples use the `@DynamoDbFlatten` annotation to effectively add all attributes of the contained `GenericRecord` class to the `Customer` class.

Standard data class

```
@DynamoDbBean
```

```
public class Customer {
    private String name;
    private GenericRecord record;

    public String getName() { return this.name; }
    public void setName(String name) { this.name = name; }

    @DynamoDbFlatten
    public GenericRecord getRecord() { return this.record; }
    public void setRecord(GenericRecord record) { this.record = record; }

@DynamoDbBean
public class GenericRecord {
    private String id;
    private String createdDate;

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

    public String getCreatedDate() { return this.createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
}
```

Lombok

```
@Data
@DynamoDbBean
public class Customer {
    private String name;
    @Getter(onMethod_=@DynamoDbFlatten)
    private GenericRecord record;
}

@Data
@DynamoDbBean
public class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdDate;
}
```

You can use the flatten annotation to flatten as many different eligible classes as you need to. The following constraints apply:

- All attribute names must be unique after they are flattened.
- There must never be more than one partition key, sort key, or table name.

Use static schemas

When you build a static table schema, use the `flatten()` method of the builder. You also supply the getter and setter methods that identify the contained class.

```
        StaticTableSchema<GenericRecord> GENERIC_RECORD_SCHEMA =
                StaticTableSchema.builder(GenericRecord.class)
                        .newItemSupplier(GenericRecord::new)
                        .addAttribute(String.class, a -> a.name("id")
                                .getter(GenericRecord::getId)
                                .setter(GenericRecord::setId)
                                .tags(primaryPartitionKey()))
                        .addAttribute(String.class, a -> a.name("created_date")
                                .getter(GenericRecord::getCreatedDate)
                                .setter(GenericRecord::setCreatedDate))
                        .build();

        StaticTableSchema<Customer> CUSTOMER_SCHEMA =
                StaticTableSchema.builder(Customer.class)
                        .newItemSupplier(Customer::new)
                        .addAttribute(String.class, a -> a.name("name")
                                .getter(Customer::getName)
                                .setter(Customer::setName))
                        // Because we are flattening a component object, we supply a
 getter and setter so the
                        // mapper knows how to access it.
                        .flatten(GENERIC_RECORD_SCHEMA, Customer::getRecord,
 Customer::setRecord)
                        .build();
```

The previous static schema example uses the following data classes.

Data classes

Standard data class

```
public class Customer {
    private String name;
    private GenericRecord record;

    public String getName() { return this.name; }
    public void setName(String name) { this.name = name; }

    public GenericRecord getRecord() { return this.record; }
    public void setRecord(GenericRecord record) { this.record = record; }

public class GenericRecord {
    private String id;
    private String createdDate;

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

    public String getCreatedDate() { return this.createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
}
```

Lombok

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

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

You can use the builder pattern to flatten as many different eligible classes as you need to.

Implications for other code

When you use the `@DynamoDbFlatten` attribute (or `flatten()` builder method), the item in DynamoDB contains an attribute for each attribute of the composed object. It also includes the attributes of the composing object.

In contrast, if you annotate a data class with a composed class and don't use `@DynamoDbFlatten`, the item is saved with the composed object as a single attribute.

For example, compare the `Customer` class shown in the flattening with composition example with and without flattening of the `record` attribute. You can visualize the difference with JSON as shown in the following table.

With flattening	Without flattening
3 attributes	2 attributes
`{ "id": "1", "createdDate": "today", "name": "my name" }`	`{ "id": "1", "record": { "createdDate": "today", "name": "my name" } }`

The difference becomes important if you have other code accessing the DynamoDB table that expects to find certain attributes.

Work with nested attributes

A nested attribute in DynamoDB is embedded in another attribute. Examples are list elements and map entries.

In Java, a DynamoDB nested attribute corresponds to a member of a class that is a `List` or `Map`. It also corresponds to an instance of a complex type, such as `Address` or `PhoneNumber`, as used in the following `Person` class.

`Person` class

```
@DynamoDbBean
```

```
public class Person {
    Integer id;
    String firstName;
    String lastName;
    Integer age;
    Map<String, Address> addresses;
    List<PhoneNumber> phoneNumbers;

    List<String> hobbies;

    @DynamoDbPartitionKey
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public Map<String, Address> getAddresses() {
```

```
        return addresses;
    }

    public void setAddresses(Map<String, Address> addresses) {
        this.addresses = addresses;
    }

    public List<PhoneNumber> getPhoneNumbers() {
        return phoneNumbers;
    }

    public void setPhoneNumbers(List<PhoneNumber> phoneNumbers) {
        this.phoneNumbers = phoneNumbers;
    }

    public List<String> getHobbies() {
        return hobbies;
    }

    public void setHobbies(List<String> hobbies) {
        this.hobbies = hobbies;
    }

    @Override
    public String toString() {
        return "Person{" +
                "id=" + id +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", age=" + age +
                ", addresses=" + addresses +
                ", phoneNumbers=" + phoneNumbers +
                ", hobbies=" + hobbies +
                '}';
    }
}
```

Address class

```
@DynamoDbBean
public class Address {
    private String street;
    private String city;
```

```
    private String state;
    private String zipCode;

    public Address() {
    }

    public String getStreet() {
        return this.street;
    }

    public String getCity() {
        return this.city;
    }

    public String getState() {
        return this.state;
    }

    public String getZipCode() {
        return this.zipCode;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public void setState(String state) {
        this.state = state;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Address address = (Address) o;
```

```
        return Objects.equals(street, address.street) && Objects.equals(city,
 address.city) && Objects.equals(state, address.state) && Objects.equals(zipCode,
 address.zipCode);
    }

    @Override
    public int hashCode() {
        return Objects.hash(street, city, state, zipCode);
    }

    @Override
    public String toString() {
        return "Address{" +
                "street='" + street + '\'' +
                ", city='" + city + '\'' +
                ", state='" + state + '\'' +
                ", zipCode='" + zipCode + '\'' +
                '}';
    }
}
```

PhoneNumber class

```
@DynamoDbBean
public class PhoneNumber {
    String type;
    String number;

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
        this.number = number;
    }
```

```
    @Override
    public String toString() {
        return "PhoneNumber{" +
                "type='" + type + '\'' +
                ", number='" + number + '\'' +
                '}';
    }
}
```

Map nested attributes

Use annotated classes

You can save nested attributes for custom classes by annotating them. The `Address` class and `PhoneNumber` class shown previously are annotated with only the `@DynamoDbBean` annotation. When the DynamoDB Enhanced Client API builds the table schema for the `Person` class with the following snippet, the API discovers the use of the `Address` and `PhoneNumber` classes and builds the corresponding mappings to work with DynamoDB.

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

Use nested schemas

The alternative approach is to use static table schema builders for each of the classes as shown in the following code.

The table schemas for the `Address` and `PhoneNumber` classes are abstract in the sense that they cannot be used with a DynamoDB table. This is because they lack definitions for the primary key. They are used, however, as nested schemas in the table schema for the `Person` class.

After comment lines 1 and 2 in the definition of `PERSON_TABLE_SCHEMA`, you see the code that uses the abstract table schemas. The use of `documentOf` in the `EnhanceType.documentOf(...)` method does not indicate that the method returns an `EnhancedDocument` type of the Enhanced Document API. The `documentOf(...)` method in this context returns an object that knows how to map its class argument to and from DynamoDB table attributes by using the table schema argument.

Static schema code

```
    // Abstract table schema that cannot be used to work with a DynamoDB table,
```

```
    // but can be used as a nested schema.
    public static final TableSchema<Address> TABLE_SCHEMA_ADDRESS =
TableSchema.builder(Address.class)
            .newItemSupplier(Address::new)
            .addAttribute(String.class, a -> a.name("street")
                    .getter(Address::getStreet)
                    .setter(Address::setStreet))
            .addAttribute(String.class, a -> a.name("city")
                    .getter(Address::getCity)
                    .setter(Address::setCity))
            .addAttribute(String.class, a -> a.name("zipcode")
                    .getter(Address::getZipCode)
                    .setter(Address::setZipCode))
            .addAttribute(String.class, a -> a.name("state")
                    .getter(Address::getState)
                    .setter(Address::setState))
            .build();

    // Abstract table schema that cannot be used to work with a DynamoDB table,
    // but can be used as a nested schema.
    public static final TableSchema<PhoneNumber> TABLE_SCHEMA_PHONENUMBER =
TableSchema.builder(PhoneNumber.class)
            .newItemSupplier(PhoneNumber::new)
            .addAttribute(String.class, a -> a.name("type")
                    .getter(PhoneNumber::getType)
                    .setter(PhoneNumber::setType))
            .addAttribute(String.class, a -> a.name("number")
                    .getter(PhoneNumber::getNumber)
                    .setter(PhoneNumber::setNumber))
            .build();

    // A static table schema that can be used with a DynamoDB table.
    // The table schema contains two nested schemas that are used to perform mapping
to/from DynamoDB.
    public static final TableSchema<Person> PERSON_TABLE_SCHEMA =
            TableSchema.builder(Person.class)
                    .newItemSupplier(Person::new)
                    .addAttribute(Integer.class, a -> a.name("id")
                            .getter(Person::getId)
                            .setter(Person::setId)
                            .addTag(StaticAttributeTags.primaryPartitionKey()))
                    .addAttribute(String.class, a -> a.name("firstName")
                            .getter(Person::getFirstName)
                            .setter(Person::setFirstName))
```

```
                    .addAttribute(String.class, a -> a.name("lastName")
                            .getter(Person::getLastName)
                            .setter(Person::setLastName))
                    .addAttribute(Integer.class, a -> a.name("age")
                            .getter(Person::getAge)
                            .setter(Person::setAge))
                    .addAttribute(EnhancedType.listOf(String.class), a ->
a.name("hobbies")
                            .getter(Person::getHobbies)
                            .setter(Person::setHobbies))
                    .addAttribute(EnhancedType.mapOf(
                            EnhancedType.of(String.class),
                            // 1. Use mapping functionality of the Address table
schema.
                            EnhancedType.documentOf(Address.class,
TABLE_SCHEMA_ADDRESS)), a -> a.name("addresses")
                            .getter(Person::getAddresses)
                            .setter(Person::setAddresses))
                    .addAttribute(EnhancedType.listOf(
                            // 2. Use mapping functionality of the PhoneNumber table
schema.
                            EnhancedType.documentOf(PhoneNumber.class,
TABLE_SCHEMA_PHONENUMBER)), a -> a.name("phoneNumbers")
                            .getter(Person::getPhoneNumbers)
                            .setter(Person::setPhoneNumbers))
                    .build();
```

Project nested attributes

For `query()` and `scan()` methods, you can specify which attributes you want to be returned in the results by using method calls such as `addNestedAttributeToProject()` and `attributesToProject()`. The DynamoDB Enhanced Client API converts the Java method call parameters into projection expressions before the request is sent.

The following example populates the `Person` table with two items, then performs three scan operations.

The first scan accesses all items in the table in order to compare the results to the other scan operations.

The second scan uses the `addNestedAttributeToProject()` builder method to return only the `street` attribute value.

The third scan operation uses the [attributesToProject()](#) builder method to return the data for the first-level attribute, `hobbies`. The attribute type of `hobbies` is a list. To access individual list items, perform a `get()` operation on the list.

```
        personDynamoDbTable = getDynamoDbEnhancedClient().table("Person",
PERSON_TABLE_SCHEMA);
        PersonUtils.createPersonTable(personDynamoDbTable, getDynamoDbClient());
        // Use a utility class to add items to the Person table.
        List<Person> personList = PersonUtils.getItemsForCount(2);
        // This utility method performs a put against DynamoDB to save the instances in
the list argument.
        PersonUtils.putCollection(getDynamoDbEnhancedClient(), personList,
personDynamoDbTable);

        // The first scan logs all items in the table to compare to the results of the
subsequent scans.
        final PageIterable<Person> allItems = personDynamoDbTable.scan();
        allItems.items().forEach(p ->
                // 1. Log what is in the table.
                logger.info(p.toString()));

        // Scan for nested attributes.
        PageIterable<Person> streetScanResult = personDynamoDbTable.scan(b -> b
                // Use the 'addNestedAttributeToProject()' or
'addNestedAttributesToProject()' to access data nested in maps in DynamoDB.
                .addNestedAttributeToProject(
                        NestedAttributeName.create("addresses", "work", "street")
                ));

        streetScanResult.items().forEach(p ->
                //2. Log the results of requesting nested attributes.
                logger.info(p.toString()));

        // Scan for a top-level list attribute.
        PageIterable<Person> phoneNumbersScanResult = personDynamoDbTable.scan(b -> b
                // Use the 'attributesToProject()' method to access first-level
attributes.
                .attributesToProject("hobbies"));

        phoneNumbersScanResult.items().forEach((p) -> {
            // 3. Log the results of the request for the 'hobbies' attribute.
            logger.info(p.toString());
```

```
            // To access an item in a list, first get the parent attribute, 'hobbies',
 then access items in the list.
            String hobby = p.getHobbies().get(1);
            // 4. Log an item in the list.
            logger.info(hobby);
        });
```

```
// Logged results from comment line 1.
Person{id=2, firstName='first name 2', lastName='last name 2', age=11,
 addresses={work=Address{street='street 21', city='city 21', state='state 21',
 zipCode='33333'}, home=Address{street='street 2', city='city 2', state='state 2',
 zipCode='22222'}}, phoneNumbers=[PhoneNumber{type='home', number='222-222-2222'},
 PhoneNumber{type='work', number='333-333-3333'}], hobbies=[hobby 2, hobby 21]}
Person{id=1, firstName='first name 1', lastName='last name 1', age=11,
 addresses={work=Address{street='street 11', city='city 11', state='state 11',
 zipCode='22222'}, home=Address{street='street 1', city='city 1', state='state 1',
 zipCode='11111'}}, phoneNumbers=[PhoneNumber{type='home', number='111-111-1111'},
 PhoneNumber{type='work', number='222-222-2222'}], hobbies=[hobby 1, hobby 11]}

// Logged results from comment line 2.
Person{id=null, firstName='null', lastName='null', age=null,
 addresses={work=Address{street='street 21', city='null', state='null',
 zipCode='null'}}, phoneNumbers=null, hobbies=null}
Person{id=null, firstName='null', lastName='null', age=null,
 addresses={work=Address{street='street 11', city='null', state='null',
 zipCode='null'}}, phoneNumbers=null, hobbies=null}

// Logged results from comment lines 3 and 4.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
 phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
hobby 21
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
 phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
hobby 11
```

Note

If the `attributesToProject()` method follows any other builder method that adds attributes that you want to project, the list of attribute names supplied to the `attributesToProject()` replaces all other attribute names.

A scan performed with the `ScanEnhancedRequest` instance in the following snippet returns only hobby data.

```
ScanEnhancedRequest lastOverwrites = ScanEnhancedRequest.builder()
        .addNestedAttributeToProject(
                NestedAttributeName.create("addresses", "work", "street"))
        .addAttributeToProject("firstName")
        // If the 'attributesToProject()' method follows other builder methods
 that add attributes for projection,
        // its list of attributes replace all previous attributes.
        .attributesToProject("hobbies")
        .build();
PageIterable<Person> hobbiesOnlyResult =
 personDynamoDbTable.scan(lastOverwrites);
hobbiesOnlyResult.items().forEach(p ->
        logger.info(p.toString()));

// Logged results.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
 phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
 phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
```

The following code snippet uses the `attributesToProject()` method first. This ordering preserves all other requested attributes.

```
ScanEnhancedRequest attributesPreserved = ScanEnhancedRequest.builder()
        // Use 'attributesToProject()' first so that the method call does not
 replace all other attributes
        // that you want to project.
        .attributesToProject("firstName")
        .addNestedAttributeToProject(
                NestedAttributeName.create("addresses", "work", "street"))
        .addAttributeToProject("hobbies")
        .build();
PageIterable<Person> allAttributesResult =
 personDynamoDbTable.scan(attributesPreserved);
allAttributesResult.items().forEach(p ->
        logger.info(p.toString()));

// Logged results.
Person{id=null, firstName='first name 2', lastName='null', age=null,
 addresses={work=Address{street='street 21', city='null', state='null',
 zipCode='null'}}, phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
```

```
Person{id=null, firstName='first name 1', lastName='null', age=null,
 addresses={work=Address{street='street 11', city='null', state='null',
 zipCode='null'}}, phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
```

Preserve empty objects with `@DynamoDbPreserveEmptyObject`

If you save a bean to Amazon DynamoDB with empty objects and you want the SDK to recreate the empty objects upon retrieval, annotate the getter of the inner bean with `@DynamoDbPreserveEmptyObject`.

To illustrate how the annotation works, the code example uses the following two beans.

Example beans

The following data class contains two `InnerBean` fields. The getter method, `getInnerBeanWithoutAnno()`, is not annotated with `@DynamoDbPreserveEmptyObject`. The `getInnerBeanWithAnno()` method is annotated.

```
@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithAnno;

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

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
    public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
 { this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

    @DynamoDbPreserveEmptyObject
    public InnerBean getInnerBeanWithAnno() { return innerBeanWithAnno; }
    public void setInnerBeanWithAnno(InnerBean innerBeanWithAnno)
 { this.innerBeanWithAnno = innerBeanWithAnno; }
```

```
    @Override
    public String toString() {
        return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
                .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
                .add("innerBeanWithAnno=" + innerBeanWithAnno)
                .add("id='" + id + "'")
                .add("name='" + name + "'")
                .toString();
    }
}
```

Instances of the following `InnerBean` class are fields of `MyBean` and are initialized as empty objects in the example code.

```
@DynamoDbBean
public class InnerBean {

    private String innerBeanField;

    public String getInnerBeanField() {
        return innerBeanField;
    }

    public void setInnerBeanField(String innerBeanField) {
        this.innerBeanField = innerBeanField;
    }

    @Override
    public String toString() {
        return "InnerBean{" +
                "innerBeanField='" + innerBeanField + '\'' +
                '}';
    }
}
```

The following code example saves a `MyBean` object with initialized inner beans to DynamoDB and then retrieves the item. The logged output shows that the `innerBeanWithoutAnno` is not initialized, but `innerBeanWithAnno` has been created.

```
    public MyBean preserveEmptyObjectAnnoUsingGetItemExample(DynamoDbTable<MyBean>
 myBeanTable) {
```

```
        // Save an item to DynamoDB.
        MyBean bean = new MyBean();
        bean.setId("1");
        bean.setInnerBeanWithoutAnno(new InnerBean());   // Instantiate the inner bean.
        bean.setInnerBeanWithAnno(new InnerBean());      // Instantiate the inner bean.
        myBeanTable.putItem(bean);

        GetItemEnhancedRequest request = GetItemEnhancedRequest.builder()
                .key(Key.builder().partitionValue("1").build())
                .build();
        MyBean myBean = myBeanTable.getItem(request);

        logger.info(myBean.toString());
        // Output 'MyBean[innerBeanWithoutAnno=null,
 innerBeanWithAnno=InnerBean{innerBeanField='null'}, id='1', name='null']'.

        return myBean;
    }
```

Alternative static schema

You can use the following `StaticTableSchema` version of the table schemas in place of the annotations on the beans.

```
    public static TableSchema<MyBean> buildStaticSchemas() {

        StaticTableSchema<InnerBean> innerBeanStaticTableSchema =
                StaticTableSchema.builder(InnerBean.class)
                        .newItemSupplier(InnerBean::new)
                        .addAttribute(String.class, a -> a.name("innerBeanField")
                                .getter(InnerBean::getInnerBeanField)
                                .setter(InnerBean::setInnerBeanField))
                        .build();

        return StaticTableSchema.builder(MyBean.class)
                .newItemSupplier(MyBean::new)
                .addAttribute(String.class, a -> a.name("id")
                        .getter(MyBean::getId)
                        .setter(MyBean::setId)
                        .addTag(primaryPartitionKey()))
                .addAttribute(String.class, a -> a.name("name")
                        .getter(MyBean::getName)
                        .setter(MyBean::setName))
```

```
                .addAttribute(EnhancedType.documentOf(InnerBean.class,
                                innerBeanStaticTableSchema),
                        a -> a.name("innerBean1")
                                .getter(MyBean::getInnerBeanWithoutAnno)
                                .setter(MyBean::setInnerBeanWithoutAnno))
                .addAttribute(EnhancedType.documentOf(InnerBean.class,
                                innerBeanStaticTableSchema,
                                b -> b.preserveEmptyObject(true)),
                        a -> a.name("innerBean2")
                                .getter(MyBean::getInnerBeanWithAnno)
                                .setter(MyBean::setInnerBeanWithAnno))
                .build();
    }
```

Avoid saving null attributes of nested objects

You can skip null attributes of nested objects when saving a data class object to DynamoDB by applying the `@DynamoDbIgnoreNulls` annotation. By contrast, top-level attributes with null values are never saved to the database.

To illustrate how the annotation works, the code example uses the following two beans.

Example beans

The following data class contains two `InnerBean` fields. The getter method, `getInnerBeanWithoutAnno()`, is not annotated. The `getInnerBeanWithIgnoreNullsAnno()` method is annotated with `@DynamoDbIgnoreNulls`.

```
@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithIgnoreNullsAnno;

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

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

```

```
    public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
    public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
 { this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

    @DynamoDbIgnoreNulls
    public InnerBean getInnerBeanWithIgnoreNullsAnno() { return
 innerBeanWithIgnoreNullsAnno; }
    public void setInnerBeanWithIgnoreNullsAnno(InnerBean innerBeanWithAnno)
 { this.innerBeanWithIgnoreNullsAnno = innerBeanWithAnno; }

    @Override
    public String toString() {
        return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
                .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
                .add("innerBeanWithIgnoreNullsAnno=" + innerBeanWithIgnoreNullsAnno)
                .add("id='" + id + "'")
                .add("name='" + name + "'")
                .toString();
    }
}
```

Instances of the following `InnerBean` class are fields of `MyBean` and are used in the following example code.

```
@DynamoDbBean
public class InnerBean {

    private String innerBeanFieldString;
    private Integer innerBeanFieldInteger;

    public String getInnerBeanFieldString() { return innerBeanFieldString; }
    public void setInnerBeanFieldString(String innerBeanFieldString)
 { this.innerBeanFieldString = innerBeanFieldString; }

    public Integer getInnerBeanFieldInteger() { return innerBeanFieldInteger; }
    public void setInnerBeanFieldInteger(Integer innerBeanFieldInteger)
 { this.innerBeanFieldInteger = innerBeanFieldInteger; }

    @Override
    public String toString() {
        return new StringJoiner(", ", InnerBean.class.getSimpleName() + "[", "]")
                .add("innerBeanFieldString='" + innerBeanFieldString + "'")
                .add("innerBeanFieldInteger=" + innerBeanFieldInteger)
```

```
                .toString();
    }
}
```

The following code example creates an `InnerBean` object and sets only one of its two attributes with a value.

```
    public void ignoreNullsAnnoUsingPutItemExample(DynamoDbTable<MyBean> myBeanTable) {
        // Create an InnerBean object and give only one attribute a value.
        InnerBean innerBeanOneAttributeSet = new InnerBean();
        innerBeanOneAttributeSet.setInnerBeanFieldInteger(200);

        // Create a MyBean instance and use the same InnerBean instance both for
 attributes.
        MyBean bean = new MyBean();
        bean.setId("1");
        bean.setInnerBeanWithoutAnno(innerBeanOneAttributeSet);
        bean.setInnerBeanWithIgnoreNullsAnno(innerBeanOneAttributeSet);

        Map<String, AttributeValue> itemMap = myBeanTable.tableSchema().itemToMap(bean,
 true);
        logger.info(itemMap.toString());
        // Log the map that is sent to the database.
        //
 {innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)}),
 id=AttributeValue(S=1),
 innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
 innerBeanFieldString=AttributeValue(NUL=true)})}

        // Save the MyBean object to the table.
        myBeanTable.putItem(bean);
    }
```

To visualize the low-level data that is sent to DynamoDB, the code logs the attribute map before saving the `MyBean` object.

The logged output shows that the `innerBeanWithIgnoreNullsAnno` outputs one attribute,

```
innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)})
```

The `innerBeanWithoutAnno` instance outputs two attributes. One attribute has a value of 200 and the other is a null-valued attribute.

```
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
 innerBeanFieldString=AttributeValue(NUL=true)})
```

JSON representation of the attribute map

The following JSON representation makes it easier to see the data that is saved to DynamoDB.

```
{
  "id": {
    "S": "1"
  },
  "innerBeanWithIgnoreNullsAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      }
    }
  },
  "innerBeanWithoutAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      },
      "innerBeanFieldString": {
        "NULL": true
      }
    }
  }
}
```

Alternative static schema

You can use the following `StaticTableSchema` version of the table schemas in place data class annotations.

```
public static TableSchema<MyBean> buildStaticSchemas() {

    StaticTableSchema<InnerBean> innerBeanStaticTableSchema =
        StaticTableSchema.builder(InnerBean.class)
            .newItemSupplier(InnerBean::new)
            .addAttribute(String.class, a -> a.name("innerBeanFieldString")
                .getter(InnerBean::getInnerBeanFieldString)
```

```
                .setter(InnerBean::setInnerBeanFieldString))
            .addAttribute(Integer.class, a -> a.name("innerBeanFieldInteger")
                .getter(InnerBean::getInnerBeanFieldInteger)
                .setter(InnerBean::setInnerBeanFieldInteger))
            .build();

    return StaticTableSchema.builder(MyBean.class)
        .newItemSupplier(MyBean::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(MyBean::getId)
            .setter(MyBean::setId)
            .addTag(primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("name")
            .getter(MyBean::getName)
            .setter(MyBean::setName))
        .addAttribute(EnhancedType.documentOf(InnerBean.class,
                innerBeanStaticTableSchema),
            a -> a.name("innerBeanWithoutAnno")
                .getter(MyBean::getInnerBeanWithoutAnno)
                .setter(MyBean::setInnerBeanWithoutAnno))
        .addAttribute(EnhancedType.documentOf(InnerBean.class,
                innerBeanStaticTableSchema,
                b -> b.ignoreNulls(true)),
            a -> a.name("innerBeanWithIgnoreNullsAnno")
                .getter(MyBean::getInnerBeanWithIgnoreNullsAnno)
                .setter(MyBean::setInnerBeanWithIgnoreNullsAnno))
        .build();
}
```

Work with JSON documents with the Enhanced Document API for DynamoDB

The Enhanced Document API for AWS SDK for Java 2.x is designed to work with document-oriented data that has no fixed schema. However, it also lets you use custom classes to map individual attributes.

The Enhanced Document API is the successor to the Document API of the AWS SDK for Java v1.x.

Contents

  - Build from a JSON string
  - Build from individual elements
- Perform CRUD operations
- Access enhanced document attributes as custom objects
- Use an EnhancedDocument without DynamoDB

Get started using the Enhanced Document API

The Enhanced Document API requires the same dependencies that are needed for the DynamoDB Enhanced Client API. It also requires a `DynamoDbEnhancedClient` instance as shown at the start of this topic.

Because the Enhanced Document API was released with version 2.20.3 of the AWS SDK for Java 2.x, you need that version or greater.

Create a `DocumentTableSchema` and a `DynamoDbTable`

To invoke commands against a DynamoDB table using the Enhanced Document API, associate the table with a client-side DynamoDbTable<EnhancedDocument> resource object.

The enhanced client's `table()` method creates a `DynamoDbTable<EnhancedDocument>` instance and requires parameters for the DynamoDB table name and a `DocumentTableSchema`.

The builder for a DocumentTableSchema requires a primary index key and one or more attribute converter providers. The `AttributeConverterProvider.defaultProvider()` method provides converters for default types. It should be specified even if you provide a custom attribute converter provider. You can add an optional secondary index key to the builder.

The following code snippet shows the code that generates the client-side representation of a DynamoDB `person` table that stores schemaless `EnhancedDocument` objects.

```
DynamoDbTable<EnhancedDocument> documentDynamoDbTable =
                enhancedClient.table("person",
                        TableSchema.documentSchemaBuilder()
                            // Specify the primary key attributes.

 .addIndexPartitionKey(TableMetadata.primaryIndexName(),"id", AttributeValueType.S)
                            .addIndexSortKey(TableMetadata.primaryIndexName(),
 "lastName", AttributeValueType.S)
```

```
                                // Specify attribute converter providers. Minimally add the
 default one.

 .attributeConverterProviders(AttributeConverterProvider.defaultProvider())
                                .build());

// Call documentTable.createTable() if "person" does not exist in DynamoDB.
// createTable() should be called only one time.
```

The following shows the JSON representation of a `person` object that is used throughout this section.

JSON `person` object

```
{
  "id": 1,
  "firstName": "Richard",
  "lastName": "Roe",
  "age": 25,
  "addresses":
    {
      "home": {
        "zipCode": "00000",
        "city": "Any Town",
        "state": "FL",
        "street": "123 Any Street"
      },
      "work": {
        "zipCode": "00001",
        "city": "Anywhere",
        "state": "FL",
        "street": "100 Main Street"
      }
    },
  "hobbies": [
    "Hobby 1",
    "Hobby 2"
  ],
  "phoneNumbers": [
    {
      "type": "Home",
      "number": "555-0100"
    },
```

```
    {
      "type": "Work",
      "number": "555-0119"
    }
  ]
}
```

Build enhanced documents

An EnhancedDocument represents a document-type object that has complex structure with nested attributes. An EnhancedDocument requires top-level attributes that match the primary key attributes specified for the DocumentTableSchema. The remaining content is arbitrary and can consist of top-level attributes and also deeply nested attributes.

You create an EnhancedDocument instance by using a builder that provides several ways to add elements.

Build from a JSON string

With a JSON string, you can build an EnhancedDocument in one method call. The following snippet creates an EnhancedDocument from a JSON string returned by the jsonPerson() helper method. The jsonPerson() method returns the JSON string version of the person object shown previously.

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

Build from individual elements

Alternatively, you can build an EnhancedDocument instance from individual components using type-safe methods of the builder.

The following example builds a person enhanced document similar to the enhanced document that is built from the JSON string in the previous example.

```
        /* Define the shape of an address map whose JSON representation looks like the
 following.
           Use 'addressMapEnhancedType' in the following EnhancedDocument.builder() to
 simplify the code.
```

```
          "home": {
            "zipCode": "00000",
            "city": "Any Town",
            "state": "FL",
            "street": "123 Any Street"
          }*/
       EnhancedType<Map<String, String>> addressMapEnhancedType =
               EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class));


       //  Use the builder's typesafe methods to add elements to the enhanced
document.
       EnhancedDocument personDocument = EnhancedDocument.builder()
               .putNumber("id", 50)
               .putString("firstName", "Shirley")
               .putString("lastName", "Rodriguez")
               .putNumber("age", 53)
               .putNull("nullAttribute")
               .putJson("phoneNumbers", phoneNumbersJSONString())
               /* Add the map of addresses whose JSON representation looks like the
following.
                       {
                         "home": {
                           "zipCode": "00000",
                           "city": "Any Town",
                           "state": "FL",
                           "street": "123 Any Street"
                         }
                       } */
               .putMap("addresses", getAddresses(), EnhancedType.of(String.class),
addressMapEnhancedType)
               .putList("hobbies", List.of("Theater", "Golf"),
EnhancedType.of(String.class))
               .build();
```

Helper methods

```
   private static String phoneNumbersJSONString() {
       return "  [" +
               "    {" +
               "      \"type\": \"Home\"," +
               "      \"number\": \"555-0140\"" +
```

```
                "    }," +
                "    {" +
                "      \"type\": \"Work\"," +
                "      \"number\": \"555-0155\"" +
                "    }" +
                "  ]";
    }

    private static Map<String, Map<String, String>> getAddresses() {
        return Map.of(
                "home", Map.of(
                        "zipCode", "00002",
                        "city", "Any Town",
                        "state", "ME",
                        "street", "123 Any Street"));

    }
```

Perform CRUD operations

After you define an `EnhancedDocument` instance, you can save it to a DynamoDB table. The following code snippet uses the personDocument that was created from individual elements.

```
documentDynamoDbTable.putItem(personDocument);
```

After you read an enhanced document instance from DynamoDB, you can extract the individual attribute values using getters as shown in the following code snippet that access the data saved from the `personDocument`. Alternatively, you can extract the complete content to a JSON string as shown in the last part of the example code.

```
        // Read the item.
        EnhancedDocument personDocFromDb =
 documentDynamoDbTable.getItem(Key.builder().partitionValue(50).build());

        // Access top-level attributes.
        logger.info("Name: {} {}", personDocFromDb.getString("firstName"),
 personDocFromDb.getString("lastName"));
        // Name: Shirley Rodriguez

        // Typesafe access of a deeply nested attribute. The addressMapEnhancedType
 shown previously defines the shape of an addresses map.
```

```
        Map<String, Map<String, String>> addresses =
 personDocFromDb.getMap("addresses", EnhancedType.of(String.class),
 addressMapEnhancedType);
        addresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));
        // {zipCode=00002, city=Any Town, street=123 Any Street, state=ME}

        // Alternatively, work with AttributeValue types checking along the way for
 deeply nested attributes.
        Map<String, AttributeValue> addressesMap =
 personDocFromDb.getMapOfUnknownType("addresses");
        addressesMap.keySet().forEach((String k) -> {
            logger.info("Looking at data for [{}] address", k);
            // Looking at data for [home] address
            AttributeValue value = addressesMap.get(k);
            AttributeValue cityValue = value.m().get("city");
            if (cityValue != null) {
                logger.info(cityValue.s());
                // Any Town
            }
        });

        List<AttributeValue> phoneNumbers =
 personDocFromDb.getListOfUnknownType("phoneNumbers");
        phoneNumbers.forEach((AttributeValue av) -> {
            if (av.hasM()) {
                AttributeValue type = av.m().get("type");
                if (type.s() != null) {
                    logger.info("Type of phone: {}", type.s());
                    // Type of phone: Home
                    // Type of phone: Work
                }
            }
        });

        String jsonPerson = personDocFromDb.toJson();
        logger.info(jsonPerson);
        // {"firstName":"Shirley","lastName":"Rodriguez","addresses":
{"home":{"zipCode":"00002","city":"Any Town","street":"123 Any
 Street","state":"ME"}},"hobbies":["Theater","Golf"],
        //     "id":50,"nullAttribute":null,"age":53,"phoneNumbers":
[{"number":"555-0140","type":"Home"},{"number":"555-0155","type":"Work"}]}
```

`EnhancedDocument` instances can be used with any method of `DynamoDbTable` or DynamoDbEnhancedClient in place of mapped data classes.

Access enhanced document attributes as custom objects

In addition to providing an API to read and write attributes with schemaless structures, the Enhanced Document API lets you convert attributes to and from instances of custom classes.

The Enhanced Document API uses `AttributeConverterProviders` and `AttributeConverters` that were shown in the control attribute conversion section as part of the DynamoDB Enhanced Client API.

In the following example, we use a `CustomAttributeConverterProvider` with its nested `AddressConverter` class to convert `Address` objects.

This example shows that you can mix data from classes and also data from structures that are built as needed. This example also shows that custom classes can be used at any level of a nested structure. The `Address` objects in this example are values used in a map.

```
    public static void attributeToAddressClassMappingExample(DynamoDbEnhancedClient
 enhancedClient, DynamoDbClient standardClient) {
        String tableName = "customer";

        // Define the DynamoDbTable for an enhanced document.
        // The schema builder provides methods for attribute converter providers and
 keys.
        DynamoDbTable<EnhancedDocument> documentDynamoDbTable =
 enhancedClient.table(tableName,
                DocumentTableSchema.builder()
                        // Add the CustomAttributeConverterProvider along with the
 default when you build the table schema.
                        .attributeConverterProviders(
                                List.of(
                                        new CustomAttributeConverterProvider(),
                                        AttributeConverterProvider.defaultProvider()))
                        .addIndexPartitionKey(TableMetadata.primaryIndexName(), "id",
 AttributeValueType.N)
                        .addIndexSortKey(TableMetadata.primaryIndexName(), "lastName",
 AttributeValueType.S)
                        .build());
        // Create the DynamoDB table if needed.
        documentDynamoDbTable.createTable();
        waitForTableCreation(tableName, standardClient);
```

```
        // The getAddressesForCustomMappingExample() helper method that provides
 'addresses' shows the use of a custom Address class
        // rather than using a Map<String, Map<String, String> to hold the address
 data.
        Map<String, Address> addresses = getAddressesForCustomMappingExample();

        // Build an EnhancedDocument instance to save an item with a mix of structures
 defined as needed and static classes.
        EnhancedDocument personDocument = EnhancedDocument.builder()
                .putNumber("id", 50)
                .putString("firstName", "Shirley")
                .putString("lastName", "Rodriguez")
                .putNumber("age", 53)
                .putNull("nullAttribute")
                .putJson("phoneNumbers", phoneNumbersJSONString())
                // Note the use of 'EnhancedType.of(Address.class)' instead of the more
 generic
                // 'EnhancedType.mapOf(EnhancedType.of(String.class),
 EnhancedType.of(String.class))' that was used in a previous example.
                .putMap("addresses", addresses, EnhancedType.of(String.class),
 EnhancedType.of(Address.class))
                .putList("hobbies", List.of("Hobby 1", "Hobby 2"),
 EnhancedType.of(String.class))
                .build();
        // Save the item to DynamoDB.
        documentDynamoDbTable.putItem(personDocument);

        // Retrieve the item just saved.
        EnhancedDocument srPerson =
 documentDynamoDbTable.getItem(Key.builder().partitionValue(50).sortValue("Rodriguez").build())

        // Access the addresses attribute.
        Map<String, Address> srAddresses = srPerson.get("addresses",
                EnhancedType.mapOf(EnhancedType.of(String.class),
 EnhancedType.of(Address.class)));

        srAddresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));

        documentDynamoDbTable.deleteTable();

// The content logged to the console shows that the saved maps were converted to
 Address instances.
```

```
Address{street='123 Main Street', city='Any Town', state='NC', zipCode='00000'}
Address{street='100 Any Street', city='Any Town', state='NC', zipCode='00000'}
```

CustomAttributeConverterProvider code

```
public class CustomAttributeConverterProvider implements AttributeConverterProvider {

    private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
 ImmutableMap.of(
            // 1. Add AddressConverter to the internal cache.
            EnhancedType.of(Address.class), new AddressConverter());

    public static CustomAttributeConverterProvider create() {
        return new CustomAttributeConverterProvider();
    }

    // 2. The enhanced client queries the provider for attribute converters if it
    //    encounters a type that it does not know how to convert.
    @SuppressWarnings("unchecked")
    @Override
    public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
        return (AttributeConverter<T>) converterCache.get(enhancedType);
    }

    // 3. Custom attribute converter
    private class AddressConverter implements AttributeConverter<Address> {
        // 4. Transform an Address object into a DynamoDB map.
        @Override
        public AttributeValue transformFrom(Address address) {

            Map<String, AttributeValue> attributeValueMap = Map.of(
                    "street", AttributeValue.fromS(address.getStreet()),
                    "city", AttributeValue.fromS(address.getCity()),
                    "state", AttributeValue.fromS(address.getState()),
                    "zipCode", AttributeValue.fromS(address.getZipCode()));

            return AttributeValue.fromM(attributeValueMap);
        }

        // 5. Transform the DynamoDB map attribute to an Address oject.
        @Override
        public Address transformTo(AttributeValue attributeValue) {
            Map<String, AttributeValue> m = attributeValue.m();
```

```
            Address address = new Address();
            address.setStreet(m.get("street").s());
            address.setCity(m.get("city").s());
            address.setState(m.get("state").s());
            address.setZipCode(m.get("zipCode").s());

            return address;
        }

        @Override
        public EnhancedType<Address> type() {
            return EnhancedType.of(Address.class);
        }

        @Override
        public AttributeValueType attributeValueType() {
            return AttributeValueType.M;
        }
    }
}
```

Address class

```
public class Address {
                  private String street;
                  private String city;
                  private String state;
                  private String zipCode;

                  public Address() {
                  }

                  public String getStreet() {
                  return this.street;
                  }

                  public String getCity() {
                  return this.city;
                  }

                  public String getState() {
                  return this.state;
                  }
```

```
                public String getZipCode() {
                return this.zipCode;
                }

                public void setStreet(String street) {
                this.street = street;
                }

                public void setCity(String city) {
                this.city = city;
                }

                public void setState(String state) {
                this.state = state;
                }

                public void setZipCode(String zipCode) {
                this.zipCode = zipCode;
                }
                }
```

Helper method that provides addresses

The following helper method provides the map that use custom `Address` instances for values rather than generic `Map<String, String>` instances for values.

```
    private static Map<String, Address> getAddressesForCustomMappingExample() {
        Address homeAddress = new Address();
        homeAddress.setStreet("100 Any Street");
        homeAddress.setCity("Any Town");
        homeAddress.setState("NC");
        homeAddress.setZipCode("00000");

        Address workAddress = new Address();
        workAddress.setStreet("123 Main Street");
        workAddress.setCity("Any Town");
        workAddress.setState("NC");
        workAddress.setZipCode("00000");

        return Map.of("home", homeAddress,
                "work", workAddress);
    }
```

Use an EnhancedDocument without DynamoDB

Although you usually use an instance of an `EnhancedDocument` to read and write document-type DynamoDB items, it can also be used independently of DynamoDB.

You can use `EnhancedDocuments` for their ability to convert between JSON strings or custom objects to low-level maps of `AttributeValues` as shown in the following example.

```
    public static void conversionWithoutDynamoDbExample() {
        Address address = new Address();
        address.setCity("my city");
        address.setState("my state");
        address.setStreet("my street");
        address.setZipCode("00000");

        // Build an EnhancedDocument instance for its conversion functionality alone.
        EnhancedDocument addressEnhancedDoc = EnhancedDocument.builder()
                // Important: You must specify attribute converter providers when you
 build an EnhancedDocument instance not used with a DynamoDB table.
                .attributeConverterProviders(new CustomAttributeConverterProvider(),
 DefaultAttributeConverterProvider.create())
                .put("addressDoc", address, Address.class)
                .build();

        // Convert address to a low-level item representation.
        final Map<String, AttributeValue> addressAsAttributeMap =
 addressEnhancedDoc.getMapOfUnknownType("addressDoc");
        logger.info("addressAsAttributeMap: {}", addressAsAttributeMap.toString());

        // Convert address to a JSON string.
        String addressAsJsonString = addressEnhancedDoc.getJson("addressDoc");
        logger.info("addressAsJsonString: {}", addressAsJsonString);
        // Convert addressEnhancedDoc back to an Address instance.
        Address addressConverted =  addressEnhancedDoc.get("addressDoc",
 Address.class);
        logger.info("addressConverted: {}", addressConverted.toString());
    }

   /* Console output:
          addressAsAttributeMap: {zipCode=AttributeValue(S=00000),
 state=AttributeValue(S=my state), street=AttributeValue(S=my street),
 city=AttributeValue(S=my city)}
```

```
        addressAsJsonString: {"zipCode":"00000","state":"my state","street":"my
street","city":"my city"}
        addressConverted: Address{street='my street', city='my city', state='my
state', zipCode='00000'}
  */
```

> **Note**
>
> When you use an enhanced document independent of a DynamoDB table, make sure you explicitly set attribute converter providers on the builder.
> In contrast, the document table schema supplies the converter providers when an enhanced document is used with a DynamoDB table.

Use extensions

The DynamoDB Enhanced Client API supports plugin extensions that provide functionality beyond mapping operations. Extensions have two hook methods, `beforeWrite()` and `afterRead()`. `beforeWrite()` modifies a write operation before it happens, and the `afterRead()` method modifies the results of a read operation after it happens. Because some operations (such as item updates) perform both a write and then a read, both hook methods are called.

Extensions are loaded in the order that they are specified in the enhanced client builder. The load order can be important because one extension can act on values that have been transformed by a previous extension.

The enhanced client API comes with a set of plugin extensions that are located in the [`extensions`] package. By default, the enhanced client loads the [`VersionedRecordExtension`] and the [`AtomicCounterExtension`]. You can override the default behavior with the enhance client builder and load any extension. You can also specify none if you don't want the default extensions.

If you load your own extensions, the enhanced client doesn't load any default extensions. If you want the behavior provided by either default extension, you need to explicitly add it to the list of extensions.

In the following example, a custom extension named `verifyChecksumExtension` is loaded after the `VersionedRecordExtension`, which is usually loaded by default by itself. The `AtomicCounterExtension` is not loaded in this example.

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

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

VersionedRecordExtension

The `VersionedRecordExtension` is loaded by default and will increment and track an item version number as items are written to the database. A condition will be added to every write that causes the write to fail if the version number of the actual persisted item doesn't match the value that the application last read. This behavior effectively provides optimistic locking for item updates. If another process updates an item between the time the first process has read the item and is writing an update to it, the write will fail.

To specify which attribute to use to track the item version number, tag a numeric attribute in the table schema.

The following snippet specifies that the `version` attribute should hold the item version number.

```
    @DynamoDbVersionAttribute
    public Integer getVersion() {...};
    public void setVersion(Integer version) {...};
```

The equivalent static table schema approach is shown in the following snippet.

```
    .addAttribute(Integer.class, a -> a.name("version")
                                       .getter(Customer::getVersion)
                                       .setter(Customer::setVersion)
                                        // Apply the 'version' tag to the attribute.

 .tags(VersionedRecordExtension.AttributeTags.versionAttribute())
```

AtomicCounterExtension

The `AtomicCounterExtension` is loaded by default and increments a tagged numerical attribute each time a record is written to the database. Start and increment values can be specified. If no values are specified, the start value is set to 0 and the attribute's value increments by 1.

To specify which attribute is a counter, tag an attribute of type `Long` in the table schema.

The following snippet shows the use of the default start and increment values for the `counter` attribute.

```
    @DynamoDbAtomicCounter
    public Long getCounter() {...};
    public void setCounter(Long counter) {...};
```

The static table schema approach is shown in the following snippet. The atomic counter extension uses a start value of 10 and increments the value by 5 each time the record is written.

```
    .addAttribute(Integer.class, a -> a.name("counter")
                                        .getter(Customer::getCounter)
                                        .setter(Customer::setCounter)
                                         // Apply the 'atomicCounter' tag to the 
attribute with start and increment values.
                                        .tags(StaticAttributeTags.atomicCounter(10L, 
5L))
```

AutoGeneratedTimestampRecordExtension

The `AutoGeneratedTimestampRecordExtension` automatically updates tagged attributes of type `Instant` with a current timestamp every time the item is successfully written to the database.

This extension is not loaded by default. Therefore, you need to specify it as a custom extension when you build the enhanced client as shown in the first example in this topic.

To specify which attribute to update with the current timestamp, tag the `Instant` attribute in the table schema.

The `lastUpdate` attribute is the target of the extensions behavior in the following snippet. Note the requirement that the attribute must be an `Instant` type.

```
    @DynamoDbAutoGeneratedTimestampAttribute
    public Instant getLastUpdate() {...}
    public void setLastUpdate(Instant lastUpdate) {...}
```

The equivalent static table schema approach is shown in the following snippet.

```
     .addAttribute(Instant.class, a -> a.name("lastUpdate")
                                         .getter(Customer::getLastUpdate)
                                         .setter(Customer::setLastUpdate)
                                         // Applying the 'autoGeneratedTimestamp' tag to
 the attribute.

 .tags(AutoGeneratedTimestampRecordExtension.AttributeTags.autoGeneratedTimestampAttribute())
```

Custom extensions

The following custom extension class shows a `beforeWrite()` method that uses an update expression. After comment line 2, we create a `SetAction` to set the `registrationDate` attribute if the item in the database doesn't already have a `registrationDate` attribute. Whenever a `Customer` object is updated, the extension makes sure that a `registrationDate` is set.

```
public final class CustomExtension implements DynamoDbEnhancedClientExtension {

    // 1. In a custom extension, use an UpdateExpression to define what action to take
 before
    //    an item is updated.
    @Override
    public WriteModification beforeWrite(DynamoDbExtensionContext.BeforeWrite context)
 {
        if ( context.operationContext().tableName().equals("Customer")
                && context.operationName().equals(OperationName.UPDATE_ITEM)) {
            return WriteModification.builder()
                    .updateExpression(createUpdateExpression())
                    .build();
        }
        return WriteModification.builder().build();  // Return an "empty"
 WriteModification instance if the extension should not be applied.
                                                      // In this case, if the code is
 not updating an item on the Customer table.
    }

    private static UpdateExpression createUpdateExpression() {
```

```
        // 2. Use a SetAction, a subclass of UpdateAction, to provide the values in the
 update.
        SetAction setAction =
                SetAction.builder()
                        .path("registrationDate")
                        .value("if_not_exists(registrationDate, :regValue)")
                        .putExpressionValue(":regValue",
 AttributeValue.fromS(Instant.now().toString()))
                        .build();
        // 3. Build the UpdateExpression with one or more UpdateAction.
        return UpdateExpression.builder()
                .addAction(setAction)
                .build();
    }
}
```

Use the DynamoDB Enhanced Client API asynchronously

If your application requires non-blocking, asynchronous calls to DynamoDB, you can use the DynamoDbEnhancedAsyncClient. It's similar to the synchronous implementation but with the following key differences:

1. When you build the `DynamoDbEnhancedAsyncClient`, you must provide the asynchronous version of the standard client, `DynamoDbAsyncClient`, as shown in the following snippet.

```
DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient.builder()
                               .dynamoDbClient(dynamoDbAsyncClient)
                               .build();
```

2. Methods that return a single data object return a `CompletableFuture` of the result instead of only the result. Your application can then do other work without having to block on the result. The following snippet shows the asynchronous `getItem()` method.

```
CompletableFuture<Customer> result = customerDynamoDbTable.getItem(customer);
// Perform other work here.
return result.join();   // Now block and wait for the result.
```

3. Methods that return paginated lists of results return an `SdkPublisher` instead of an `SdkIterable` that the synchronous `DynamoDbEnhanceClient` returns for the same methods.

Your application can then subscribe a handler to that publisher to deal with the results asynchronously without having to block.

```
PagePublisher<Customer> results = customerDynamoDbTable.query(r ->
 r.queryConditional(keyEqualTo(k -> k.partitionValue("Smith"))));
results.subscribe(myCustomerResultsProcessor);
// Perform other work and let the processor handle the results asynchronously.
```

For a more complete example of working with the `SdkPublisher API`, see the example in the section that discusses the asynchronous `scan()` method of this guide.

Data class annotations

The following table lists the annotations that can be used on data classes and provides links to information and examples in this guide. The table is sorted in ascending alphabetical order by annotation name.

Data class annotations used in this guide

Annotation name	Annotation applies to[1]	What it does	Where it is shown in this guide
DynamoDbAtomicCounter	attribute[2]	Increments a tagged numerical attribute each time a record is written to the database.	Introduction and discussion.
DynamoDbAttribute	attribute	Defines or renames a bean property that is mapped to a DynamoDB table attribute.	• Initial discussion. • Get started section —see Note. • In MovieActor class In Query method examples.
DynamoDbAutoGeneratedTimestampAttribute	attribute	Updates a tagged attribute with a current timestamp	Introduction and discussion.

Annotation name	Annotation applies to[1]	What it does	Where it is shown in this guide
		every time the item is successfully written to the database	
DynamoDbBean	class	Marks a data class as mappable to a table schema.	First use on the Customer class in the Get started section. Several usages appear throughout the guide.
DynamoDbConvertedBy	attribute	Associates a custom `AttributeConverter` with the annotated attribute.	Initial discussion and example.
DynamoDbFlatten	attribute	Flattens all the attributes of a separate DynamoDB data class and adds them as top-level attributes to the record that is read and written to the database.	• Initial discussion. • Implications for other code.
DynamoDbIgnore	attribute	Results in the attribute remaining unmapped.	• Initial discussion. • Use in the ProductCatalog class.
DynamoDbIgnoreNulls	attribute	Prevents saving null attributes of nested DynamoDb objects.	Discussion and examples.

Annotation name	Annotation applies to[1]	What it does	Where it is shown in this guide
DynamoDbImmutable	class	Marks an immutable data class as mappable to a table schema.	• Introduction to the annotation. • Use in the ProductCatalog class. • Use with Lombok.
DynamoDbPartitionKey	attribute	Marks an attribute as the primary partition key (hash key) of the DynamoDb table.	• Initial usage on the Customer class in the Get started section. • With Lombok.
DynamoDbPreserveEmptyObject	attribute	Specifies that if no data is present for the object mapped to the annotated attribute, the object should be initialized with all null fields.	Discussion and examples.
DynamoDbSecondaryPartitionKey	attribute	Marks an attribute as a partition key for a global secondary index.	• Use in secondary indices and example. • In Query method examples. • In Lombok example • With immutable classes.

Annotation name	Annotation applies to[1]	What it does	Where it is shown in this guide
DynamoDbS econdarySortKey	attribute	Marks an attribute as an optional sort key for a global or local secondary index.	• Use in secondary indices and example. • In Query method examples. • In Lombok example. • With immutable classes.
DynamoDbSortKey	attribute	Marks an attribute as the optional primary sort key (range key).	• Get started section on Customer class. • With immutable classes. • In Lombok example. • In Query method examples.
DynamoDbU pdateBehavior	attribute	Specifies the behavior when this attribute is updated as part of an 'update' operation such as UpdateItem.	Introduction and example.
DynamoDbVersionAtt ribute	attribute	Increments an item version number.	Introduction and discussion.

[1]You can apply an attribute-level annotations to the getter or setter, but not both. This guide shows annotations on getters.

[2]The term `property` is normally used for a value encapsulated in a JavaBean data class. However, this guide uses the term `attribute` instead, to be consistent with the terminology used by DynamoDB.

Work with Amazon EC2

This section provides examples of programming Amazon EC2 that use the AWS SDK for Java 2.x.

Topics

Manage Amazon EC2 instances

Create an instance

Create a new Amazon EC2 instance by calling the Ec2Client's `runInstances` method, providing it with a RunInstancesRequest containing the Amazon Machine Image (AMI) to use and an instance type.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
    public static String createEC2Instance(Ec2Client ec2,String name, String amiId ) {

        RunInstancesRequest runRequest = RunInstancesRequest.builder()
```

```
                .imageId(amiId)
                .instanceType(InstanceType.T1_MICRO)
                .maxCount(1)
                .minCount(1)
                .build();

        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceId = response.instances().get(0).instanceId();

        Tag tag = Tag.builder()
                .key("Name")
                .value(name)
                .build();

        CreateTagsRequest tagRequest = CreateTagsRequest.builder()
                .resources(instanceId)
                .tags(tag)
                .build();

        try {
            ec2.createTags(tagRequest);
            System.out.printf(
                    "Successfully started EC2 Instance %s based on AMI %s",
                    instanceId, amiId);

          return instanceId;

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

See the complete example on GitHub.

Start an instance

To start an Amazon EC2 instance, call the Ec2Client’s `startInstances` method, providing it with a StartInstancesRequest containing the ID of the instance to start.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

Code

```
    public static void startInstance(Ec2Client ec2, String instanceId) {

        StartInstancesRequest request = StartInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        ec2.startInstances(request);
        System.out.printf("Successfully started instance %s", instanceId);
    }
```

See the complete example on GitHub.

Stop an instance

To stop an Amazon EC2 instance, call the Ec2Client's `stopInstances` method, providing it with a StopInstancesRequest containing the ID of the instance to stop.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

Code

```
    public static void stopInstance(Ec2Client ec2, String instanceId) {

        StopInstancesRequest request = StopInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        ec2.stopInstances(request);
        System.out.printf("Successfully stopped instance %s", instanceId);
```

```
    }
```

See the complete example on GitHub.

Reboot an instance

To reboot an Amazon EC2 instance, call the Ec2Client's `rebootInstances` method, providing it with a RebootInstancesRequest containing the ID of the instance to reboot.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.RebootInstancesRequest;
```

Code

```
    public static void rebootEC2Instance(Ec2Client ec2, String instanceId) {

      try {
            RebootInstancesRequest request = RebootInstancesRequest.builder()
                .instanceIds(instanceId)
                    .build();

            ec2.rebootInstances(request);
            System.out.printf(
                "Successfully rebooted instance %s", instanceId);
    } catch (Ec2Exception e) {
          System.err.println(e.awsErrorDetails().errorMessage());
          System.exit(1);
     }
  }
```

See the complete example on GitHub.

Describe instances

To list your instances, create a DescribeInstancesRequest and call the Ec2Client's `describeInstances` method. It will return a DescribeInstancesResponse object that you can use to list the Amazon EC2 instances for your account and region.

Instances are grouped by *reservation*. Each reservation corresponds to the call to `startInstances` that launched the instance. To list your instances, you must first call the `DescribeInstancesResponse` class' `reservations` method, and then call `instances` on each returned Reservation object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
    public static void describeEC2Instances( Ec2Client ec2){

        String nextToken = null;

        try {

            do {
                DescribeInstancesRequest request =
 DescribeInstancesRequest.builder().maxResults(6).nextToken(nextToken).build();
                DescribeInstancesResponse response = ec2.describeInstances(request);

                for (Reservation reservation : response.reservations()) {
                    for (Instance instance : reservation.instances()) {
                        System.out.println("Instance Id is " + instance.instanceId());
                        System.out.println("Image id is "+  instance.imageId());
                        System.out.println("Instance type is "+
 instance.instanceType());
                        System.out.println("Instance state name is "+
 instance.state().name());
                        System.out.println("monitoring information is "+
 instance.monitoring().state());

                }
            }
                nextToken = response.nextToken();
```

```
            } while (nextToken != null);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Results are paged; you can get further results by passing the value returned from the result object's `nextToken` method to a new request object's `nextToken` method, then using the new request object in your next call to `describeInstances`.

See the complete example on GitHub.

Monitor an instance

You can monitor various aspects of your Amazon EC2 instances, such as CPU and network utilization, available memory, and disk space remaining. To learn more about instance monitoring, see Monitoring Amazon EC2 in the Amazon EC2 User Guide for Linux Instances.

To start monitoring an instance, you must create a MonitorInstancesRequest with the ID of the instance to monitor, and pass it to the Ec2Client's `monitorInstances` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

Code

```
    public static void monitorInstance( Ec2Client ec2, String instanceId) {

        MonitorInstancesRequest request = MonitorInstancesRequest.builder()
                .instanceIds(instanceId).build();

        ec2.monitorInstances(request);
        System.out.printf(
                "Successfully enabled monitoring for instance %s",
                instanceId);
```

```
    }
```

See the complete example on GitHub.

Stop instance monitoring

To stop monitoring an instance, create an UnmonitorInstancesRequest with the ID of the instance to stop monitoring, and pass it to the Ec2Client’s `unmonitorInstances` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

Code

```
    public static void unmonitorInstance(Ec2Client ec2, String instanceId) {
        UnmonitorInstancesRequest request = UnmonitorInstancesRequest.builder()
                .instanceIds(instanceId).build();

        ec2.unmonitorInstances(request);

        System.out.printf(
                "Successfully disabled monitoring for instance %s",
                instanceId);
    }
```

See the complete example on GitHub.

More information

- RunInstances in the Amazon EC2 API Reference
- DescribeInstances in the Amazon EC2 API Reference
- StartInstances in the Amazon EC2 API Reference
- StopInstances in the Amazon EC2 API Reference
- RebootInstances in the Amazon EC2 API Reference
- MonitorInstances in the Amazon EC2 API Reference

- UnmonitorInstances in the Amazon EC2 API Reference

Use AWS Regions and Availability Zones

Describe Regions

To list the Regions available to your account, call the Ec2Client's `describeRegions` method. It returns a DescribeRegionsResponse. Call the returned object's `regions` method to get a list of Region objects that represent each Region.

Imports

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

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

See the complete example on GitHub.

Describe availability zones

To list each Availability Zone available to your account, call the Ec2Client's `describeAvailabilityZones` method. It returns a DescribeAvailabilityZonesResponse. Call its `availabilityZones` method to get a list of AvailabilityZone objects that represent each Availability Zone.

Imports

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

Create the Ec2Client.

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

Then call describeAvailabilityZones() and retrieve results.

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

See the complete example on GitHub.

Describe accounts

To list EC2-related information about your account, call the Ec2Client's `describeAccountAttributes` method. This method returns a DescribeAccountAttributesResponse object. Invoke this objects `accountAttributes` method to get a list of AccountAttribute objects. You can iterate through the list to retrieve an AccountAttribute object.

You can get your account's attribute values by invoking the AccountAttribute object's attributeValues method. This method returns a list of AccountAttributeValue objects. You can iterate through this second list to display the value of attributes (see the following code example).

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccount {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
                .region(region)
                .build();

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

    public static void describeEC2Account(Ec2Client ec2) {
        try {
```

```
            DescribeAccountAttributesResponse accountResults =
 ec2.describeAccountAttributes();
            accountResults.accountAttributes().forEach(attribute -> {
                System.out.print("\n The name of the attribute is " +
 attribute.attributeName());
                attribute.attributeValues().forEach(
                        myValue -> System.out.print("\n The value of the attribute is "
 + myValue.attributeValue()));
            });

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

See the complete example on GitHub.

More information

- Regions and Availability Zones in the Amazon EC2 User Guide for Linux Instances
- DescribeRegions in the Amazon EC2 API Reference
- DescribeAvailabilityZones in the Amazon EC2 API Reference

Work with security groups in Amazon EC2

Create a security group

To create a security group, call the Ec2Client's `createSecurityGroup` method with a CreateSecurityGroupRequest that contains the key's name.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
```

```
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

Code

```
            CreateSecurityGroupRequest createRequest =
 CreateSecurityGroupRequest.builder()
                .groupName(groupName)
                .description(groupDesc)
                .vpcId(vpcId)
                .build();

            CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
```

See the complete example on GitHub.

Configure a security group

A security group can control both inbound (ingress) and outbound (egress) traffic to your Amazon EC2 instances.

To add ingress rules to your security group, use the Ec2Client's `authorizeSecurityGroupIngress` method, providing the name of the security group and the access rules (IpPermission) you want to assign to it within an AuthorizeSecurityGroupIngressRequest object. The following example shows how to add IP permissions to a security group.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

Code

First, create an Ec2Client

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

Then use the Ec2Client's `authorizeSecurityGroupIngress` method,

```
            IpRange ipRange = IpRange.builder()
                .cidrIp("0.0.0.0/0").build();

            IpPermission ipPerm = IpPermission.builder()
                .ipProtocol("tcp")
                .toPort(80)
                .fromPort(80)
                .ipRanges(ipRange)
                .build();

            IpPermission ipPerm2 = IpPermission.builder()
                .ipProtocol("tcp")
                .toPort(22)
                .fromPort(22)
                .ipRanges(ipRange)
                .build();

            AuthorizeSecurityGroupIngressRequest authRequest =
                AuthorizeSecurityGroupIngressRequest.builder()
                        .groupName(groupName)
                        .ipPermissions(ipPerm, ipPerm2)
                        .build();

            AuthorizeSecurityGroupIngressResponse authResponse =
            ec2.authorizeSecurityGroupIngress(authRequest);

            System.out.printf(
                "Successfully added ingress policy to Security Group %s",
                groupName);

            return resp.groupId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return "";
    }
```

To add an egress rule to the security group, provide similar data in an AuthorizeSecurityGroupEgressRequest to the Ec2Client's `authorizeSecurityGroupEgress` method.

See the complete example on GitHub.

Describe security groups

To describe your security groups or get information about them, call the Ec2Client's `describeSecurityGroups` method. It returns a DescribeSecurityGroupsResponse that you can use to access the list of security groups by calling its `securityGroups` method, which returns a list of SecurityGroup objects.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
     public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {

        try {
            DescribeSecurityGroupsRequest request =
                DescribeSecurityGroupsRequest.builder()
                        .groupIds(groupId).build();

            DescribeSecurityGroupsResponse response =
                ec2.describeSecurityGroups(request);

             for(SecurityGroup group : response.securityGroups()) {
                System.out.printf(
                    "Found Security Group with id %s, " +
                            "vpc id %s " +
```

```
                            "and description %s",
                    group.groupId(),
                    group.vpcId(),
                    group.description());
            }
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Delete a security group

To delete a security group, call the Ec2Client's `deleteSecurityGroup` method, passing it a DeleteSecurityGroupRequest that contains the ID of the security group to delete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
    public static void deleteEC2SecGroup(Ec2Client ec2,String groupId) {

        try {
            DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
                .groupId(groupId)
                .build();

            ec2.deleteSecurityGroup(request);
            System.out.printf(
                "Successfully deleted Security Group with id %s", groupId);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
```

See the complete example on GitHub.

More information

- Amazon EC2 Security Groups in the Amazon EC2 User Guide for Linux Instances
- Authorize inbound traffic for Your Linux Instances in the Amazon EC2 User Guide for Linux Instances
- CreateSecurityGroup in the Amazon EC2 API Reference
- DescribeSecurityGroups in the Amazon EC2 API Reference
- DeleteSecurityGroup in the Amazon EC2 API Reference
- AuthorizeSecurityGroupIngress in the Amazon EC2 API Reference

Work with Amazon EC2 instance metadata

A Java SDK client for the Amazon EC2 Instance Metadata Service (metadata client) allows your applications to access metadata on their local EC2 instance. The metadata client works with the local instance of IMDSv2 (Instance Metadata Service v2) and uses session-oriented requests.

Two client classes are available in the SDK. The synchronous `Ec2MetadataClient` is for blocking operations, and the `Ec2MetadataAsyncClient` is for asynchronous, non-blocking use cases.

Get started

To use the metadata client, add the `imds` Maven artifact to your project. You also need classes for an `SdkHttpClient` (or an `SdkAsyncHttpClient` for the asynchronous variant) on the classpath.

The following Maven XML shows dependency snippets for using the synchronous UrlConnectionHttpClient along with the dependency for metadata clients.

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>VERSION</version>
            <type>pom</type>
            <scope>import</scope>
```

```
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>imds</artifactId>
    </dependency>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>url-connection-client</artifactId>
    </dependency>
    <!-- other dependencies -->
</dependencies>
```

Search the Maven central repository for the latest version of the bom artifact.

To use an asynchronous HTTP client, replace the dependency snippet for the `url-connection-client` artifact. For example, the following snippet brings in the NettyNioAsyncHttpClient implementation.

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

Use the metadata client

Instantiate a metadata client

You can instantiate an instance of a synchronous `Ec2MetadataClient` when only one implementation of the `SdkHttpClient` interface is present on the classpath. To do so, call the static `Ec2MetadataClient#create()` method as shown in the following snippet.

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

If your application has multiple implementations of the `SdkHttpClient` or `SdkHttpAsyncClient` interface, you must specify an implementation for the metadata client to use as shown in the the section called "Configurable HTTP client" section.

> **Note**
>
> For most service clients, such as Amazon S3, the SDK for Java automatically adds implementations of the `SdkHttpClient` or `SdkHttpAsyncClient` interface. If your metadata client uses the same implementation, then `Ec2MetadataClient#create()` will work. If you require a different implementation, you must specify it when you create the metadata client.

Send requests

To retrieve instance metadata, instantiate the `EC2MetadataClient` class and call the `get` method with a path parameter that specifies the [instance metadata category](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instancedata-data-categories.html).

The following example prints the value associated with the `ami-id` key to the console.

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

If the path isn't valid, the `get` method throws an exception.

Reuse the same client instance for multiple requests, but call `close` on the client when it is no longer needed to release resources. After the close method is called, the client instance can't be used anymore.

Parse responses

EC2 instance metadata can be output in different formats. Plain text and JSON are the most commonly used formats. The metadata clients offer ways to work with those formats.

As the following example shows, use the `asString` method to get the data as a Java string. You can also use the `asList` method to separate a plain text response that returns multiple lines.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/");
String fullResponse = response.asString();
List<String> splits = response.asList();
```

If the response is in JSON, use the `Ec2MetadataResponse#asDocument` method to parse the JSON response into a Document instance as shown in the following code snippet.

```
Document fullResponse = response.asDocument();
```

An exception will be thrown if the format of the metadata is not in JSON. If the response is successfully parsed, you can use the document API to inspect the response in more detail. Consult the instance metadata category chart to learn which metadata categories deliver JSON-formatted responses.

Configure a metadata client

Retries

You can configure a metadata client with a retry mechanism. If you do, then the client can automatically retry requests that fail for unexpected reasons. By default, the client retries three times on a failed request with an exponential backoff time between attempts.

If your use case requires a different retry mechanism, you can customize the client using the `retryPolicy` method on its builder. For example, the following example shows a synchronous client configured with a fixed delay of two seconds between attempts and five retry attempts.

```
BackoffStrategy fixedBackoffStrategy =
 FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                     .retryPolicy(retryPolicyBuilder ->
 retryPolicyBuilder.numRetries(5)

 .backoffStrategy(fixedBackoffStrategy))
                     .build();
```

There are several BackoffStrategies that you can use with a metadata client.

You can also disable the retry mechanism entirely, as the following snippet shows.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                    .retryPolicy(Ec2MetadataRetryPolicy.none())
                    .build();
```

Using `Ec2MetadataRetryPolicy#none()` disables the default retry policy so that the metadata client attempts no retries.

IP version

By default, a metadata client uses the IPV4 endpoint at `http://169.254.169.254`. To change the client to use the IPV6 version, use either the `endpointMode` or the `endpoint` method of the builder. An exception results if both methods are called on the builder.

The following examples show both IPV6 options.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                     .endpointMode(EndpointMode.IPV6)
                     .build();
```

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                     .endpoint(URI.create("http://[fd00:ec2::254]"))
                     .build();
```

Key features

Asynchronous client

To use the non-blocking version of the client, instantiate an instance of the `Ec2MetadataAsyncClient` class. The code in the following example creates an asynchronous client with default settings and uses the `get` method to retrieve the value for the `ami-id` key.

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

The `java.util.concurrent.CompletableFuture` returned by the `get` method completes when the response returns. The following example prints the `ami-id` metadata to the console.

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

Configurable HTTP client

The builder for each metadata client has a `httpClient` method that you can use to supply a customized HTTP client.

The following example shows code for a custom `UrlConnectionHttpClient` instance.

```
SdkHttpClient httpClient =
    UrlConnectionHttpClient.builder()
                            .socketTimeout(Duration.ofMinutes(5))
                            .proxyConfiguration(proxy ->
 proxy.endpoint(URI.create("http://proxy.example.net:8888"))))
                            .build();
Ec2MetadataClient metaDataClient =
    Ec2MetadataClient.builder()
                      .httpClient(httpClient)
                      .build();
// Use the metaDataClient instance.
metaDataClient.close();   // Close the instance when no longer needed.
```

The following example shows code for a custom `NettyNioAsyncHttpClient` instance with an asynchronous metadata client.

```
SdkAsyncHttpClient httpAsyncClient =
    NettyNioAsyncHttpClient.builder()
                            .connectionTimeout(Duration.ofMinutes(5))
                            .maxConcurrency(100)
                            .build();
Ec2MetadataAsyncClient asyncMetaDataClient =
    Ec2MetadataAsyncClient.builder()
                           .httpClient(httpAsyncClient)
                           .build();
// Use the asyncMetaDataClient instance.
asyncMetaDataClient.close();   // Close the instance when no longer needed.
```

The the section called “HTTP clients” topic in this guide provides details on how to configure the HTTP clients that are available in the SDK for Java.

Token caching

Because the metadata clients use IMDSv2, all requests are associated with a session. A session is defined by a token that has an expiration, which the metadata client manages for you. Every metadata request automatically reuses the token until it expires.

By default, a token lasts for six hours (21,600 seconds). We recommend that you keep the default time-to-live value, unless your specific use case requires advanced configuration.

If needed, configure the duration by using the `tokenTtl` builder method. For example, the code in the following snippet creates a client with a session duration of five minutes.

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

If you omit calling the `tokenTtl` method on the builder, the default duration of 21,600 is used instead.

Work with IAM

This section provides examples of programming AWS Identity and Access Management (IAM) by using the AWS SDK for Java 2.x.

AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources. For a complete guide to IAM, visit the IAM User Guide.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- Work with IAM server certificates

Manage IAM access keys

Create an access key

To create an IAM access key, call the `IamClient’s` `createAccessKey` method with a `CreateAccessKeyRequest` object.

> **Note**
>
> You must set the region to **AWS_GLOBAL** for `IamClient` calls to work because IAM is a global service.

Imports

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static String createIAMAccessKey(IamClient iam,String user) {

        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                .userName(user).build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
           String keyId = response.accessKey().accessKeyId();
           return keyId;

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

```
    }
```

See the complete example on GitHub.

List access keys

To list the access keys for a given user, create a `ListAccessKeysRequest` object that contains the user name to list keys for, and pass it to the `IamClient’s` `listAccessKeys` method.

> **Note**
>
> If you do not supply a user name to `listAccessKeys`, it will attempt to list access keys associated with the AWS account that signed the request.

Imports

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
    public static void listKeys( IamClient iam,String userName ){

        try {
            boolean done = false;
            String newMarker = null;

            while (!done) {
                ListAccessKeysResponse response;

            if(newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                        .userName(userName).build();
                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
```

```
                    .userName(userName)
                    .marker(newMarker).build();
                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata :
                    response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
                        metadata.accessKeyId());
            }

            if (!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }

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

The results of `listAccessKeys` are paged (with a default maximum of 100 records per call). You can call `isTruncated` on the returned `ListAccessKeysResponse` object to see if the query returned fewer results then are available. If so, then call `marker` on the `ListAccessKeysResponse` and use it when creating a new request. Use that new request in the next invocation of `listAccessKeys`.

See the complete example on GitHub.

Retrieve an access key's last used time

To get the time an access key was last used, call the `IamClient's` `getAccessKeyLastUsed` method with the access key's ID (which can be passed in using a `GetAccessKeyLastUsedRequest` object.

You can then use the returned `GetAccessKeyLastUsedResponse` object to retrieve the key's last used time.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedRequest;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void getAccessKeyLastUsed(IamClient iam, String accessId ){

        try {
            GetAccessKeyLastUsedRequest request = GetAccessKeyLastUsedRequest.builder()
                    .accessKeyId(accessId).build();

            GetAccessKeyLastUsedResponse response = iam.getAccessKeyLastUsed(request);

            System.out.println("Access key was last used at: " +
                    response.accessKeyLastUsed().lastUsedDate());

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
```

See the complete example on GitHub.

Activate or deactivate access keys

You can activate or deactivate an access key by creating an `UpdateAccessKeyRequest` object, providing the access key ID, optionally the user name, and the desired `status`, then passing the request object to the `IamClient’s` `updateAccessKey` method.

Imports

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
    public static void updateKey(IamClient iam, String username, String accessId,
String status ) {

        try {
            if (status.toLowerCase().equalsIgnoreCase("active")) {
                statusType = StatusType.ACTIVE;
            } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                statusType = StatusType.INACTIVE;
            } else {
                statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
            }
            UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
              .accessKeyId(accessId)
              .userName(username)
              .status(statusType)
              .build();

            iam.updateAccessKey(request);

            System.out.printf(
              "Successfully updated the status of access key %s to" +
                      "status %s for user %s", accessId, status, username);

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

See the complete example on GitHub.

Delete an access key

To permanently delete an access key, call the `IamClient’s` `deleteKey` method, providing it with a `DeleteAccessKeyRequest` containing the access key’s ID and username.

> **Note**
>
> Once deleted, a key can no longer be retrieved or used. To temporarily deactivate a key so that it can be activated again later, use `updateAccessKey` method instead.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void deleteKey(IamClient iam ,String username, String accessKey ) {

        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                " from user " + username);

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

See the complete example on GitHub.

More information

- CreateAccessKey in the IAM API Reference
- ListAccessKeys in the IAM API Reference
- GetAccessKeyLastUsed in the IAM API Reference
- UpdateAccessKey in the IAM API Reference
- DeleteAccessKey in the IAM API Reference

Manage IAM Users

Create a User

Create a new IAM user by providing the user name to the IamClient's `createUser` method using a CreateUserRequest object containing the user name.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;
```

Code

```
    public static String createIAMUser(IamClient iam, String username ) {

        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();

            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            CreateUserResponse response = iam.createUser(request);

            // Wait until the user is created
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
 iamWaiter.waitUntilUserExists(userRequest);
            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user().userName();
```

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

See the complete example on GitHub.

List Users

To list the IAM users for your account, create a new ListUsersRequest and pass it to the IamClient's `listUsers` method. You can retrieve the list of users by calling `users` on the returned ListUsersResponse object.

The list of users returned by `listUsers` is paged. You can check to see there are more results to retrieve by calling the response object's `isTruncated` method. If it returns `true`, then call the response object's `marker()` method. Use the marker value to create a new request object. Then call the `listUsers` method again with the new request.

Imports

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.services.iam.model.User;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
    public static void listAllUsers(IamClient iam ) {

        try {

             boolean done = false;
             String newMarker = null;

             while(!done) {
                ListUsersResponse response;

                if (newMarker == null) {
```

```
                    ListUsersRequest request = ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                        .marker(newMarker).build();
                    response = iam.listUsers(request);
                }

                for(User user : response.users()) {
                 System.out.format("\n Retrieved user %s", user.userName());
                }

                if(!response.isTruncated()) {
                  done = true;
                } else {
                    newMarker = response.marker();
                }
            }
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Update a User

To update a user, call the IamClient object's `updateUser` method, which takes a UpdateUserRequest object that you can use to change the user's *name* or *path*.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;
```

Code

```
    public static void updateIAMUser(IamClient iam, String curName,String newName ) {

        try {
```

```
            UpdateUserRequest request = UpdateUserRequest.builder()
                    .userName(curName)
                    .newUserName(newName)
                    .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s",
                newName);
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Delete a User

To delete a user, call the IamClient's `deleteUser` request with a UpdateUserRequest object set with the user name to delete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void deleteIAMUser(IamClient iam, String userName) {

        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
```

See the complete example on GitHub.

More Information

- IAM Users in the IAM User Guide
- Managing IAM Users in the IAM User Guide
- CreateUser in the IAM API Reference
- ListUsers in the IAM API Reference
- UpdateUser in the IAM API Reference
- DeleteUser in the IAM API Reference

Create IAM policies with the AWS SDK for Java 2.x

The IAM Policy Builder API is a library that you can use to build IAM policies in Java and upload them to AWS Identity and Access Management (IAM).

Instead of building an IAM policy by manually assembling a JSON string or by reading a file, the API provides a client-side, object-oriented approach to generate the JSON string. When you read an existing IAM policy in JSON format, the API converts it to an IamPolicy instance for handling.

The IAM Policy Builder API became available with version 2.20.105 of the SDK, so use that version or a later one in your Maven build file. The latest version number of the SDK is listed on Maven central.

The following snippet shows an example dependency block for a Maven `pom.xml` file. This allows you to use the IAM Policy Builder API in your project.

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>iam-policy-builder</artifactId>
    <version>2.20.139</version>
</dependency>
```

Create an `IamPolicy`

This section shows several examples of how to build policies by using the IAM Policy Builder API.

In each of the following examples, start with the `IamPolicy.Builder` and add one or more statements by using the `addStatement` method. Following this pattern, the IamStatement.Builder has methods to add the effect, actions, resources, and conditions to the statement.

Example: Create a time-based policy

The following example creates an identity-based policy that permits the Amazon DynamoDB `GetItem` action between two points in time.

```
    public String timeBasedPolicyExample() {
        IamPolicy policy = IamPolicy.builder()
                .addStatement(b -> b
                        .effect(IamEffect.ALLOW)
                        .addAction("dynamodb:GetItem")
                        .addResource(IamResource.ALL)
                        .addCondition(b1 -> b1
                                .operator(IamConditionOperator.DATE_GREATER_THAN)
                                .key("aws:CurrentTime")
                                .value("2020-04-01T00:00:00Z"))
                        .addCondition(b1 -> b1
                                .operator(IamConditionOperator.DATE_LESS_THAN)
                                .key("aws:CurrentTime")
                                .value("2020-06-30T23:59:59Z")))
                .build();

        // Use an IamPolicyWriter to write out the JSON string to a more readable
 format.
        return policy.toJson(IamPolicyWriter.builder()
                        .prettyPrint(true)
                        .build());
    }
```

JSON output

The last statement in the previous example returns the following JSON string.

Read more about this example in the *AWS Identity and Access Management User Guide*.

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
```

```
    "Action" : "dynamodb:GetItem",
    "Resource" : "*",
    "Condition" : {
      "DateGreaterThan" : {
        "aws:CurrentTime" : "2020-04-01T00:00:00Z"
      },
      "DateLessThan" : {
        "aws:CurrentTime" : "2020-06-30T23:59:59Z"
      }
    }
  }
```

Example: Specify multiple conditions

The following example shows how you can create an identity-based policy that allows access to specific DynamoDB attributes. The policy contains two conditions.

```
    public String multipleConditionsExample() {
        IamPolicy policy = IamPolicy.builder()
                .addStatement(b -> b
                        .effect(IamEffect.ALLOW)
                        .addAction("dynamodb:GetItem")
                        .addAction("dynamodb:BatchGetItem")
                        .addAction("dynamodb:Query")
                        .addAction("dynamodb:PutItem")
                        .addAction("dynamodb:UpdateItem")
                        .addAction("dynamodb:DeleteItem")
                        .addAction("dynamodb:BatchWriteItem")
                        .addResource("arn:aws:dynamodb:*:*:table/table-name")

 .addConditions(IamConditionOperator.STRING_EQUALS.addPrefix("ForAllValues:"),
                                "dynamodb:Attributes",
                                List.of("column-name1", "column-name2", "column-
name3"))
                        .addCondition(b1 ->
 b1.operator(IamConditionOperator.STRING_EQUALS.addSuffix("IfExists"))
                                .key("dynamodb:Select")
                                .value("SPECIFIC_ATTRIBUTES")))
                .build();

        return policy.toJson(IamPolicyWriter.builder()
                .prettyPrint(true).build());
    }
```

JSON output

The last statement in the previous example returns the following JSON string.

Read more about this example in the *AWS Identity and Access Management User Guide*.

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : [ "dynamodb:GetItem", "dynamodb:BatchGetItem", "dynamodb:Query",
 "dynamodb:PutItem", "dynamodb:UpdateItem", "dynamodb:DeleteItem",
 "dynamodb:BatchWriteItem" ],
    "Resource" : "arn:aws:dynamodb:*:*:table/table-name",
    "Condition" : {
      "ForAllValues:StringEquals" : {
        "dynamodb:Attributes" : [ "column-name1", "column-name2", "column-name3" ]
      },
      "StringEqualsIfExists" : {
        "dynamodb:Select" : "SPECIFIC_ATTRIBUTES"
      }
    }
  }
```

Example: Specify principals

The following example shows how to create a resource-based policy that denies access to a bucket for all principals except for those specified in the condition.

```
    public String specifyPrincipalsExample() {
        IamPolicy policy = IamPolicy.builder()
                .addStatement(b -> b
                        .effect(IamEffect.DENY)
                        .addAction("s3:*")
                        .addPrincipal(IamPrincipal.ALL)
                        .addResource("arn:aws:s3:::BUCKETNAME/*")
                        .addResource("arn:aws:s3:::BUCKETNAME")
                        .addCondition(b1 -> b1
                                .operator(IamConditionOperator.ARN_NOT_EQUALS)
                                .key("aws:PrincipalArn")
                                .value("arn:aws:iam::444455556666:user/user-name")))
                .build();
        return policy.toJson(IamPolicyWriter.builder()
```

```
                .prettyPrint(true).build());
    }
```

JSON output

The last statement in the previous example returns the following JSON string.

Read more about this example in the *AWS Identity and Access Management User Guide*.

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Deny",
    "Principal" : "*",
    "Action" : "s3:*",
    "Resource" : [ "arn:aws:s3:::BUCKETNAME/*", "arn:aws:s3:::BUCKETNAME" ],
    "Condition" : {
      "ArnNotEquals" : {
        "aws:PrincipalArn" : "arn:aws:iam::444455556666:user/user-name"
      }
    }
  }
}
```

Example: Allow cross-account access

The following example shows how to allow another AWS account to upload objects to your bucket while retaining full owner control of the uploaded objects.

```
    public String allowCrossAccountAccessExample() {
        IamPolicy policy = IamPolicy.builder()
                .addStatement(b -> b
                        .effect(IamEffect.ALLOW)
                        .addPrincipal(IamPrincipalType.AWS, "111122223333")
                        .addAction("s3:PutObject")
                        .addResource("arn:aws:s3:::DOC-EXAMPLE-BUCKET/*")
                        .addCondition(b1 -> b1
                                .operator(IamConditionOperator.STRING_EQUALS)
                                .key("s3:x-amz-acl")
                                .value("bucket-owner-full-control")))
                .build();
        return policy.toJson(IamPolicyWriter.builder()
```

```
                 .prettyPrint(true).build());
     }
```

JSON output

The last statement in the previous example returns the following JSON string.

Read more about this example in the *Amazon Simple Storage Service User Guide*.

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "111122223333"
    },
    "Action" : "s3:PutObject",
    "Resource" : "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
    "Condition" : {
      "StringEquals" : {
        "s3:x-amz-acl" : "bucket-owner-full-control"
      }
    }
  }
}
```

Use an `IamPolicy` with IAM

After you have created a `IamPolicy` instance, you use an `IamClient` to work with the IAM service.

The following example builds a policy that allows an IAM identity to write items to a DynamoDB table in the account that is specified with the `accountID` parameter. The policy is then uploaded to IAM as a JSON string.

```
    public String createAndUploadPolicyExample(IamClient iam, String accountID, String
 policyName) {
        // Build the policy.
        IamPolicy policy =
                IamPolicy.builder() // 'version' defaults to "2012-10-17".
                        .addStatement(IamStatement.builder()
                                .effect(IamEffect.ALLOW)
```

```
                                .addAction("dynamodb:PutItem")
                                .addResource("arn:aws:dynamodb:us-east-1:" + accountID
+ ":table/exampleTableName")
                                .build())
                        .build();
        // Upload the policy.
        iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
        return policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
    }
```

The next example builds on the previous example. The code downloads the policy and uses it as the basis for a new policy by copying and altering the statement. The new policy is then uploaded.

```
    public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName, String newPolicyName) {

        String policyArn = "arn:aws:iam::" + accountID + ":policy/" + policyName;
        GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

        String policyVersion = getPolicyResponse.policy().defaultVersionId();
        GetPolicyVersionResponse getPolicyVersionResponse =
                iam.getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

        // Create an IamPolicy instance from the JSON string returned from IAM.
        String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
StandardCharsets.UTF_8);
        IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

            /*
             All IamPolicy components are immutable, so use the copy method that
creates a new instance that
             can be altered in the same method call.

             Add the ability to get an item from DynamoDB as an additional action.
            */
        IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

        // Create a new statement that replaces the original statement.
```

```
        IamPolicy newPolicy = policy.copy(p ->
 p.statements(Arrays.asList(newStatement)));

        // Upload the new policy. IAM now has both policies.
        iam.createPolicy(r -> r.policyName(newPolicyName)
                .policyDocument(newPolicy.toJson()));

        return newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
    }
```

IamClient

The previous examples use an `IamClient` argument that is created as shown in the following snippet.

```
IamClient iam = IamClient.builder().region(Region.AWS_GLOBAL).build();
```

Policies in JSON

The examples return the following JSON strings.

```
First example
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : "dynamodb:PutItem",
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}

Second example
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : [ "dynamodb:PutItem", "dynamodb:GetItem" ],
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}
```

Work with IAM policies

Create a policy

To create a new policy, provide the policy's name and a JSON-formatted policy document in a CreatePolicyRequest to the IamClient's `createPolicy` method.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
```

Code

```
    public static String createIAMPolicy(IamClient iam, String policyName ) {

        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
                .policyName(policyName)
                .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);

            // Wait until the policy is created
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
  iamWaiter.waitUntilPolicyExists(polRequest);
            waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();
```

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

See the complete example on GitHub.

Get a policy

To retrieve an existing policy, call the IamClient's `getPolicy` method, providing the policy's ARN within a GetPolicyRequest object.

Imports

```
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void getIAMPolicy(IamClient iam, String policyArn) {

        try {
            GetPolicyRequest request = GetPolicyRequest.builder()
                .policyArn(policyArn).build();

            GetPolicyResponse response = iam.getPolicy(request);
            System.out.format("Successfully retrieved policy %s",
                response.policy().policyName());

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

See the complete example on GitHub.

Attach a role policy

You can attach a policy to an IAM role by calling the IamClient's `attachRolePolicy` method, providing it with the role name and policy ARN in an AttachRolePolicyRequest.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

Code

```
   public static void attachIAMRolePolicy(IamClient iam, String roleName, String
 policyArn ) {

        try {

             ListAttachedRolePoliciesRequest request =
 ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse  response =
 iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

            // Ensure that the policy is not attached to this role
            String polArn = "";
            for (AttachedPolicy policy: attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn)==0) {
                   System.out.println(roleName +
                            " policy is already attached to this role.");
                    return;
                }
          }
```

```
            AttachRolePolicyRequest attachRequest =
                AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policyArn)
                        .build();

            iam.attachRolePolicy(attachRequest);

            System.out.println("Successfully attached policy " + policyArn +
                " to role " + roleName);

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

     System.out.println("Done");
    }
```

See the complete example on GitHub.

List attached role policies

List attached policies on a role by calling the IamClient's `listAttachedRolePolicies` method. It takes a ListAttachedRolePoliciesRequest object that contains the role name to list the policies for.

Call `getAttachedPolicies` on the returned ListAttachedRolePoliciesResponse object to get the list of attached policies. Results may be truncated; if the `ListAttachedRolePoliciesResponse` object's `isTruncated` method returns `true`, call the `ListAttachedRolePoliciesResponse` object's `marker` method. Use the marker returned to create a new request and use it to call `listAttachedRolePolicies` again to get the next batch of results.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

Code

```
   public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

       try {

            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                   .roleName(roleName)
                   .build();

           ListAttachedRolePoliciesResponse  response =
iam.listAttachedRolePolicies(request);
           List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

           // Ensure that the policy is not attached to this role
           String polArn = "";
           for (AttachedPolicy policy: attachedPolicies) {
               polArn = policy.policyArn();
               if (polArn.compareTo(policyArn)==0) {
                  System.out.println(roleName +
                           " policy is already attached to this role.");
                   return;
               }
         }

           AttachRolePolicyRequest attachRequest =
               AttachRolePolicyRequest.builder()
                       .roleName(roleName)
                       .policyArn(policyArn)
                       .build();

           iam.attachRolePolicy(attachRequest);

           System.out.println("Successfully attached policy " + policyArn +
               " to role " + roleName);

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

     System.out.println("Done");
```

```
    }
```

See the complete example on GitHub.

Detach a role policy

To detach a policy from a role, call the IamClient's `detachRolePolicy` method, providing it with the role name and policy ARN in a DetachRolePolicyRequest.

Imports

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void detachPolicy(IamClient iam, String roleName, String policyArn )
 {

        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                " from role " + roleName);

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

See the complete example on GitHub.

More information

- Overview of IAM Policies in the IAM User Guide.

- AWS IAM Policy Reference in the IAM User Guide.
- CreatePolicy in the IAM API Reference
- GetPolicy in the IAM API Reference
- AttachRolePolicy in the IAM API Reference
- ListAttachedRolePolicies in the IAM API Reference
- DetachRolePolicy in the IAM API Reference

Work with IAM server certificates

To enable HTTPS connections to your website or application on AWS, you need an SSL/TLS *server certificate*. You can use a server certificate provided by AWS Certificate Manager or one that you obtained from an external provider.

We recommend that you use ACM to provision, manage, and deploy your server certificates. With ACM you can request a certificate, deploy it to your AWS resources, and let ACM handle certificate renewals for you. Certificates provided by ACM are free. For more information about ACM, see the AWS Certificate Manager User Guide.

Get a server certificate

You can retrieve a server certificate by calling the IamClient's `getServerCertificate` method, passing it a GetServerCertificateRequest with the certificate's name.

Imports

```
import software.amazon.awssdk.services.iam.model.GetServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.GetServerCertificateResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void getCertificate(IamClient iam,String certName ) {

        try {
            GetServerCertificateRequest request = GetServerCertificateRequest.builder()
                    .serverCertificateName(certName)
```

```
                    .build();

            GetServerCertificateResponse response = iam.getServerCertificate(request);
            System.out.format("Successfully retrieved certificate with body %s",
                response.serverCertificate().certificateBody());

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

See the complete example on GitHub.

List server certificates

To list your server certificates, call the IamClient's `listServerCertificates` method with a ListServerCertificatesRequest. It returns a ListServerCertificatesResponse.

Call the returned `ListServerCertificateResponse` object's `serverCertificateMetadataList` method to get a list of ServerCertificateMetadata objects that you can use to get information about each certificate.

Results may be truncated; if the `ListServerCertificateResponse` object's `isTruncated` method returns `true`, call the `ListServerCertificatesResponse` object's `marker` method and use the marker to create a new request. Use the new request to call `listServerCertificates` again to get the next batch of results.

Imports

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesRequest;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesResponse;
import software.amazon.awssdk.services.iam.model.ServerCertificateMetadata;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
    public static void listCertificates(IamClient iam) {

        try {
```

```
            boolean done = false;
            String newMarker = null;

            while(!done) {
              ListServerCertificatesResponse response;

            if (newMarker == null) {
                ListServerCertificatesRequest request =
                        ListServerCertificatesRequest.builder().build();
                response = iam.listServerCertificates(request);
            } else {
                ListServerCertificatesRequest request =
                        ListServerCertificatesRequest.builder()
                                .marker(newMarker).build();
                response = iam.listServerCertificates(request);
            }

            for(ServerCertificateMetadata metadata :
                    response.serverCertificateMetadataList()) {
                System.out.printf("Retrieved server certificate %s",
                        metadata.serverCertificateName());
            }

            if(!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }

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

See the complete example on GitHub.

Update a server certificate

You can update a server certificate's name or path by calling the IamClient's `updateServerCertificate` method. It takes a UpdateServerCertificateRequest object set with the server certificate's current name and either a new name or new path to use.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateResponse;
```

Code

```
    public static void updateCertificate(IamClient iam, String curName, String newName)
 {

        try {
            UpdateServerCertificateRequest request =
                UpdateServerCertificateRequest.builder()
                        .serverCertificateName(curName)
                        .newServerCertificateName(newName)
                        .build();

            UpdateServerCertificateResponse response =
                iam.updateServerCertificate(request);


            System.out.printf("Successfully updated server certificate to name %s",
                newName);

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

See the complete example on GitHub.

Delete a server certificate

To delete a server certificate, call the IamClient's `deleteServerCertificate` method with a DeleteServerCertificateRequest containing the certificate's name.

Imports

```
import software.amazon.awssdk.services.iam.model.DeleteServerCertificateRequest;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void deleteCert(IamClient iam,String certName ) {

        try {
            DeleteServerCertificateRequest request =
                DeleteServerCertificateRequest.builder()
                        .serverCertificateName(certName)
                        .build();

            iam.deleteServerCertificate(request);
            System.out.println("Successfully deleted server certificate " +
                    certName);

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

See the complete example on GitHub.

More information

- Working with Server Certificates in the IAM User Guide
- GetServerCertificate in the IAM API Reference
- ListServerCertificates in the IAM API Reference
- UpdateServerCertificate in the IAM API Reference
- DeleteServerCertificate in the IAM API Reference
- AWS Certificate Manager User Guide

Work with Kinesis

This section provides examples of programming Amazon Kinesis using the AWS SDK for Java 2.x.

For more information about Kinesis, see the Amazon Kinesis Developer Guide.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- Subscribe to Amazon Kinesis Data Streams

Subscribe to Amazon Kinesis Data Streams

The following examples show you how to retrieve and process data from Amazon Kinesis Data Streams using the `subscribeToShard` method. Kinesis Data Streams now employs the enhanced fanout feature and a low-latency HTTP/2 data retrieval API, making it easier for developers to run multiple low-latency, high-performance applications on the same Kinesis Data Stream.

Set up

First, create an asynchronous Kinesis client and a SubscribeToShardRequest object. These objects are used in each of the following examples to subscribe to Kinesis events.

Imports

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicInteger;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEventStream;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponse;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
```

Code

```
        Region region = Region.US_EAST_1;
        KinesisAsyncClient client = KinesisAsyncClient.builder()
        .region(region)
```

```
        .build();

        SubscribeToShardRequest request = SubscribeToShardRequest.builder()
                .consumerARN(CONSUMER_ARN)
                .shardId("arn:aws:kinesis:us-east-1:111122223333:stream/
StockTradeStream")
                .startingPosition(s -> s.type(ShardIteratorType.LATEST)).build();
```

Use the builder interface

You can use the `builder` method to simplify the creation of the SubscribeToShardResponseHandler.

Using the builder, you can set each lifecycle callback with a method call instead of implementing the full interface.

Code

```
    private static CompletableFuture<Void> responseHandlerBuilder(KinesisAsyncClient
 client, SubscribeToShardRequest request) {
        SubscribeToShardResponseHandler responseHandler =
 SubscribeToShardResponseHandler
                .builder()
                .onError(t -> System.err.println("Error during stream - " +
 t.getMessage()))
                .onComplete(() -> System.out.println("All records stream
 successfully"))
                // Must supply some type of subscriber
                .subscriber(e -> System.out.println("Received event - " + e))
                .build();
        return client.subscribeToShard(request, responseHandler);
    }
```

For more control of the publisher, you can use the `publisherTransformer` method to customize the publisher.

Code

```
    private static CompletableFuture<Void>
 responseHandlerBuilderPublisherTransformer(KinesisAsyncClient client,
 SubscribeToShardRequest request) {
        SubscribeToShardResponseHandler responseHandler =
 SubscribeToShardResponseHandler
```

```
                .builder()
                .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
                .publisherTransformer(p -> p.filter(e -> e instanceof
SubscribeToShardEvent).limit(100))
                .subscriber(e -> System.out.println("Received event - " + e))
                .build();
        return client.subscribeToShard(request, responseHandler);
    }
```

See the complete example on GitHub.

Use a custom response handler

For full control of the subscriber and publisher, implement the `SubscribeToShardResponseHandler` interface.

In this example, you implement the `onEventStream` method, which allows you full access to the publisher. This demonstrates how to transform the publisher to event records for printing by the subscriber.

Code

```
    private static CompletableFuture<Void>
responseHandlerBuilderClassic(KinesisAsyncClient client, SubscribeToShardRequest
request) {
        SubscribeToShardResponseHandler responseHandler = new
SubscribeToShardResponseHandler() {

            @Override
            public void responseReceived(SubscribeToShardResponse response) {
                System.out.println("Receieved initial response");
            }

            @Override
            public void onEventStream(SdkPublisher<SubscribeToShardEventStream>
publisher) {
                publisher
                        // Filter to only SubscribeToShardEvents
                        .filter(SubscribeToShardEvent.class)
                        // Flat map into a publisher of just records
                        .flatMapIterable(SubscribeToShardEvent::records)
                        // Limit to 1000 total records
```

```
                        .limit(1000)
                        // Batch records into lists of 25
                        .buffer(25)
                        // Print out each record batch
                        .subscribe(batch -> System.out.println("Record Batch - " +
batch));
            }

            @Override
            public void complete() {
                System.out.println("All records stream successfully");
            }

            @Override
            public void exceptionOccurred(Throwable throwable) {
                System.err.println("Error during stream - " + throwable.getMessage());
            }
        };
        return client.subscribeToShard(request, responseHandler);
    }
```

See the complete example on GitHub.

Use the visitor interface

You can use a Visitor object to subscribe to specific events you're interested in watching.

Code

```
    private static CompletableFuture<Void>
responseHandlerBuilderVisitorBuilder(KinesisAsyncClient client,
SubscribeToShardRequest request) {
        SubscribeToShardResponseHandler.Visitor visitor =
SubscribeToShardResponseHandler.Visitor
                .builder()
                .onSubscribeToShardEvent(e -> System.out.println("Received subscribe to
shard event " + e))
                .build();
        SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
                .builder()
                .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
```

```
                .subscriber(visitor)
                .build();
        return client.subscribeToShard(request, responseHandler);
    }
```

See the complete example on GitHub.

Use a custom subscriber

You can also implement your own custom subscriber to subscribe to the stream.

This code snippet shows an example subscriber.

Code

```
    private static class MySubscriber implements
 Subscriber<SubscribeToShardEventStream> {

        private Subscription subscription;
        private AtomicInteger eventCount = new AtomicInteger(0);

        @Override
        public void onSubscribe(Subscription subscription) {
            this.subscription = subscription;
            this.subscription.request(1);
        }

        @Override
        public void onNext(SubscribeToShardEventStream shardSubscriptionEventStream) {
            System.out.println("Received event " + shardSubscriptionEventStream);
            if (eventCount.incrementAndGet() >= 100) {
                // You can cancel the subscription at any time if you wish to stop
 receiving events.
                subscription.cancel();
            }
            subscription.request(1);
        }

        @Override
        public void onError(Throwable throwable) {
            System.err.println("Error occurred while stream - " +
 throwable.getMessage());
        }
```

```
        @Override
        public void onComplete() {
            System.out.println("Finished streaming all events");
        }
    }
```

You can pass the custom subscriber to the `subscribe` method as shown in the following code snippet.

Code

```
    private static CompletableFuture<Void>
responseHandlerBuilderSubscriber(KinesisAsyncClient client, SubscribeToShardRequest
request) {
        SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
                .builder()
                .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
                .subscriber(MySubscriber::new)
                .build();
        return client.subscribeToShard(request, responseHandler);
    }
```

See the complete example on GitHub.

Write data records into a Kinesis data stream

You can use the KinesisClient object to write data records into a Kinesis data stream by using the `putRecords` method. To successfully invoke this method, create a PutRecordsRequest object. You pass the name of the data stream to the `streamName` method. Also you must pass the data by using the `putRecords` method (as shown in the following code example).

Imports

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
```

```
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
```

In the following Java code example, notice that **StockTrade** object is used as the data to write to the Kinesis data stream. Before running this example, ensure that you have created the data stream.

Code

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream to which records are
 written (for example, StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
```

```
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        // Ensure that the Kinesis Stream is valid.
        validateStream(kinesisClient, streamName);
        setStockData(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void setStockData(KinesisClient kinesisClient, String streamName) {
        try {
            // Repeatedly send stock trades with a 100 milliseconds wait in between.
            StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

            // Put in 50 Records for this example.
            int index = 50;
            for (int x = 0; x < index; x++) {
                StockTrade trade = stockTradeGenerator.getRandomTrade();
                sendStockTrade(trade, kinesisClient, streamName);
                Thread.sleep(100);
            }

        } catch (KinesisException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }

    private static void sendStockTrade(StockTrade trade, KinesisClient kinesisClient,
            String streamName) {
        byte[] bytes = trade.toJsonAsBytes();

        // The bytes could be null if there is an issue with the JSON serialization by
        // the Jackson JSON library.
        if (bytes == null) {
            System.out.println("Could not get JSON bytes for stock trade");
            return;
        }

        System.out.println("Putting trade: " + trade);
        PutRecordRequest request = PutRecordRequest.builder()
```

```
                .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as
 the partition key, explained in
                                                              // the Supplemental Information
 section below.
                .streamName(streamName)
                .data(SdkBytes.fromByteArray(bytes))
                .build();

        try {
            kinesisClient.putRecord(request);
        } catch (KinesisException e) {
            System.err.println(e.getMessage());
        }
    }

    private static void validateStream(KinesisClient kinesisClient, String streamName)
 {
        try {
            DescribeStreamRequest describeStreamRequest =
 DescribeStreamRequest.builder()
                    .streamName(streamName)
                    .build();

            DescribeStreamResponse describeStreamResponse =
 kinesisClient.describeStream(describeStreamRequest);

            if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
 {
                System.err.println("Stream " + streamName + " is not active. Please
 wait a few moments and try again.");
                System.exit(1);
            }

        } catch (KinesisException e) {
            System.err.println("Error found while describing the stream " +
 streamName);
            System.err.println(e);
            System.exit(1);
        }
    }
}
```

See the complete example on GitHub.

Use a third-party library

You can use other third-party libraries instead of implementing a custom subscriber. This example demonstrates using the RxJava implementation, but you can use any library that implements the Reactive Streams interfaces. See the RxJava wiki page on Github for more information on that library.

To use the library, add it as a dependency. If you're using Maven, the example shows the POM snippet to use.

POM Entry

```
<dependency>
 <groupId>io.reactivex.rxjava2</groupId>
 <artifactId>rxjava</artifactId>
 <version>2.1.14</version>
</dependency>
```

Imports

```
import java.net.URI;
import java.util.concurrent.CompletableFuture;

import io.reactivex.Flowable;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.http.Protocol;
import software.amazon.awssdk.http.SdkHttpConfigurationOption;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.StartingPosition;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
import software.amazon.awssdk.utils.AttributeMap;
```

This example uses RxJava in the `onEventStream` lifecycle method. This gives you full access to the publisher, which can be used to create an Rx Flowable.

Code

```
        SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
            .builder()
            .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
            .onEventStream(p -> Flowable.fromPublisher(p)
                                        .ofType(SubscribeToShardEvent.class)

.flatMapIterable(SubscribeToShardEvent::records)
                                        .limit(1000)
                                        .buffer(25)
                                        .subscribe(e -> System.out.println("Record
batch = " + e)))
            .build();
```

You can also use the `publisherTransformer` method with the `Flowable` publisher. You must adapt the `Flowable` publisher to an *SdkPublisher*, as shown in the following example.

Code

```
        SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
            .builder()
            .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
            .publisherTransformer(p ->
SdkPublisher.adapt(Flowable.fromPublisher(p).limit(100)))
            .build();
```

See the complete example on GitHub.

More information

- SubscribeToShardEvent in the Amazon Kinesis API Reference
- SubscribeToShard in the Amazon Kinesis API Reference

Invoke, list, and delete AWS Lambda functions

This section provides examples of programming with the Lambda service client by using the AWS SDK for Java 2.x.

Topics

- Invoke a Lambda function
- List Lambda functions
- Delete a Lambda function

Invoke a Lambda function

You can invoke a Lambda function by creating a `LambdaClient` object and invoking its `invoke` method. Create an `InvokeRequest` object to specify additional information such as the function name and the payload to pass to the Lambda function. Function names appear as *arn:aws:lambda:us-east-1:123456789012:function:HelloFunction*. You can retrieve the value by looking at the function in the AWS Management Console.

To pass payload data to a function, create a `SdkBytes` object that contains information. For example, in the following code example, notice the JSON data passed to the Lambda function.

Imports

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

Code

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

```
    public static void invokeFunction(LambdaClient awsLambda, String functionName) {

         InvokeResponse res = null ;
        try {
```

```
            //Need a SdkBytes instance for the payload
            String json = "{\"Hello \":\"Paris\"}";
            SdkBytes payload = SdkBytes.fromUtf8String(json) ;

            //Setup an InvokeRequest
            InvokeRequest request = InvokeRequest.builder()
                    .functionName(functionName)
                    .payload(payload)
                    .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String() ;
            System.out.println(value);

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

List Lambda functions

Build a `LambdaClient` object and invoke its `listFunctions` method. This method returns a `ListFunctionsResponse` object. You can invoke this object's `functions` method to return a list of `FunctionConfiguration` objects. You can iterate through the list to retrieve information about the functions. For example, the following Java code example shows how to get each function name.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;
```

Code

The following Java code example demonstrates how to retrieve a list of function names.

```
    public static void listFunctions(LambdaClient awsLambda) {

        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();

            for (FunctionConfiguration config: list) {
                System.out.println("The function name is "+config.functionName());
            }

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Delete a Lambda function

Build a `LambdaClient` object and invoke its `deleteFunction` method. Create a `DeleteFunctionRequest` object and pass it to the `deleteFunction` method. This object contains information such as the name of the function to delete. Function names appear as *arn:aws:lambda:us-east-1:123456789012:function:HelloFunction*. You can retrieve the value by looking at the function in the AWS Management Console.

Imports

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

Code

The following Java code demonstrates how to delete a Lambda function.

```
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName ) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
```

```
                    .functionName(functionName)
                    .build();

            awsLambda.deleteFunction(request);
            System.out.println("The "+functionName +" function was deleted");

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Work with Amazon S3

This section provides examples of programming with Amazon Simple Storage Service (S3) using the AWS SDK for Java 2.x.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Note

From version 2.18.x and onward, the AWS SDK for Java 2.x uses virtual host-style addressing when including an endpoint override. This applies as long as the bucket name is a valid DNS label.

Call the `forcePathStyle` method with `true` in your client builder to force the client to use path-style addressing for buckets.

The following example shows a service client configured with an endpoint override and using path-style addressing.

```
S3Client client = S3Client.builder()
                          .region(Region.US_WEST_2)
                          .endpointOverride(URI.create("https://s3.us-west-2.amazonaws.com"))
                          .forcePathStyle(true)
                          .build();
```

Use access points or Multi-Region Access Points

After Amazon S3 access points or Multi-Region Access Points are set up, you can call object methods, such as `putObject` and `getObject` and provide the access point identifier instead of a bucket name.

For example, if an access point ARN identifier is `arn:aws:s3:us-west-2:123456789012:accesspoint/test`, you can use the following snippet to call the `putObject` method.

```
        Path path = Paths.get(URI.create("file:///temp/file.txt"));

        s3Client.putObject(builder -> builder
                        .key("myKey")
                        .bucket("arn:aws:s3:us-west-2:123456789012:accesspoint/test")
                , path);
```

In place of the ARN string, you can also use the bucket-style alias of the access point for the `bucket` parameter.

To use Multi-Region Access Point, replace the `bucket` parameter with the Multi-Region Access Point ARN that has the following format.

```
arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias
```

Add the following Maven dependency to work with Multi-Region Access Points using the SDK for Java. Search maven central for the latest version.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>auth-crt</artifactId>
  <version>VERSION</version>
</dependency>
```

Topics

- Create, list, and delete Amazon S3 buckets
- Work with Amazon S3 objects
- Work with Amazon S3 pre-signed URLs

- Cross-Region access for Amazon S3
- Amazon S3 checksums with AWS SDK for Java
- Use a performant S3 client: AWS CRT-based S3 client
- Transfer files and directories with the Amazon S3 Transfer Manager

Create, list, and delete Amazon S3 buckets

Every object (file) in Amazon S3 must reside within a *bucket*. A bucket represents a collection (container) of objects. Each bucket must have a unique *key* (name). For detailed information about buckets and their configuration, see Working with Amazon S3 Buckets in the Amazon Simple Storage Service User Guide.

> **Note**
>
> Best Practice
> We recommend that you enable the AbortIncompleteMultipartUpload lifecycle rule on your Amazon S3 buckets.
> This rule directs Amazon S3 to abort multipart uploads that don't complete within a specified number of days after being initiated. When the set time limit is exceeded, Amazon S3 aborts the upload and then deletes the incomplete upload data.
> For more information, see Lifecycle Configuration for a Bucket with Versioning in the Amazon Simple Storage Service User Guide.

> **Note**
>
> These code snippets assume that you understand the material in basics, and have configured default AWS credentials using the information in the section called "Setup for single sign-on access for the SDK".

Create a bucket

Build a CreateBucketRequest and provide a bucket name. Pass it to the S3Client's `createBucket` method. Use the `S3Client` to do additional operations such as listing or deleting buckets as shown in later examples.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

First create an S3Client.

```
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
```

Make a Create Bucket Request.

```
 import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketOps {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        String bucket = "bucket" + System.currentTimeMillis();
        System.out.println(bucket);
        createBucket(s3, bucket);
        performOperations(s3, bucket);
    }

    // Create a bucket by using a S3Waiter object
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
 s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

List buckets

Build a ListBucketsRequest. Use the S3Client's `listBuckets` method to retrieve the list of buckets. If the request succeeds a ListBucketsResponse is returned. Use this response object to retrieve the list of buckets.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

First create an S3Client.

```
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
```

Make a List Buckets Request.

```
        // List buckets
        ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder().build();
        ListBucketsResponse listBucketsResponse = s3.listBuckets(listBucketsRequest);
        listBucketsResponse.buckets().stream().forEach(x ->
 System.out.println(x.name()));
```

See the complete example on GitHub.

Delete a bucket

Before you can delete an Amazon S3 bucket, you must ensure that the bucket is empty or the service will return an error. If you have a versioned bucket, you must also delete any versioned objects that are in the bucket.

Topics

- Delete objects in a bucket
- Delete an empty bucket

Delete objects in a bucket

Build a ListObjectsV2Request and use the S3Client's `listObjects` method to retrieve the list of objects in the bucket. Then use the `deleteObject` method on each object to delete it.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
```

Code

First create an S3Client.

```
        ProfileCredentialsProvider credentialsProvider =
 ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .credentialsProvider(credentialsProvider)
                .build();
```

Delete all objects in the bucket.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketDeletion {
    public static void main(String[] args) throws Exception {
        final String usage = """

                Usage:
                    <bucket>

                Where:
                    bucket - The bucket to delete (for example, bucket1).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteObjectsInBucket(s3, bucket);
        s3.close();
    }

    public static void deleteObjectsInBucket(S3Client s3, String bucket) {
        try {
```

```
            // To delete a bucket, all the objects in the bucket must be deleted first.
            ListObjectsV2Request listObjectsV2Request = ListObjectsV2Request.builder()
                    .bucket(bucket)
                    .build();
            ListObjectsV2Response listObjectsV2Response;

            do {
                listObjectsV2Response = s3.listObjectsV2(listObjectsV2Request);
                for (S3Object s3Object : listObjectsV2Response.contents()) {
                    DeleteObjectRequest request = DeleteObjectRequest.builder()
                            .bucket(bucket)
                            .key(s3Object.key())
                            .build();
                    s3.deleteObject(request);
                }
            } while (listObjectsV2Response.isTruncated());
            DeleteBucketRequest deleteBucketRequest =
 DeleteBucketRequest.builder().bucket(bucket).build();
            s3.deleteBucket(deleteBucketRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

See the complete example on GitHub.

Delete an empty bucket

Build a DeleteBucketRequest with a bucket name and pass it to the S3Client's `deleteBucket` method.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

```
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

First create an S3Client.

```
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
```

Delete the bucket.

```
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucket)
                .build();

        s3.deleteBucket(deleteBucketRequest);
        s3.close();
```

See the complete example on GitHub.

Work with Amazon S3 objects

An Amazon S3 object represents a file or collection of data. Every object must be contained in a bucket.

Note

Best Practice
We recommend that you enable the AbortIncompleteMultipartUpload lifecycle rule on your Amazon S3 buckets.
This rule directs Amazon S3 to abort multipart uploads that don't complete within a specified number of days after being initiated. When the set time limit is exceeded, Amazon S3 aborts the upload and then deletes the incomplete upload data.

For more information, see Lifecycle Configuration for a Bucket with Versioning in the Amazon Simple Storage Service User Guide.

Note

These code snippets assume that you understand the material in basics, and have configured default AWS credentials using the information in the section called "Setup for single sign-on access for the SDK".

Topics

- Upload an object
- Upload objects in multiple parts
- Delete an object
- List objects
- More examples

Upload an object

Build a PutObjectRequest and supply a bucket name and key name. Then use the S3Client's `putObject` method with a RequestBody that contains the object content and the `PutObjectRequest` object. *The bucket must exist, or the service will return an error.*

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
```

```
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
                Region region = Region.US_WEST_2;
                s3 = S3Client.builder()
                                .region(region)
                                .build();

                createBucket(s3, bucketName, region);

                PutObjectRequest objectRequest = PutObjectRequest.builder()
                                .bucket(bucketName)
                                .key(key)
                                .build();

                s3.putObject(objectRequest,
 RequestBody.fromByteBuffer(getRandomByteBuffer(10_000)));
```

See the complete example on GitHub.

Upload objects in multiple parts

Use the S3Client’s `createMultipartUpload` method to get an upload ID. Then use the `uploadPart` method to upload each part. Finally, use the S3Client’s `completeMultipartUpload` method to tell Amazon S3 to merge all the uploaded parts and finish the upload operation.

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
                // First create a multipart upload and get the upload id
                CreateMultipartUploadRequest createMultipartUploadRequest = 
 CreateMultipartUploadRequest.builder()
                                .bucket(bucketName)
                                .key(key)
                                .build();

                CreateMultipartUploadResponse response = 
 s3.createMultipartUpload(createMultipartUploadRequest);
                String uploadId = response.uploadId();
                System.out.println(uploadId);

                // Upload all the different parts of the object
```

```
                UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
                                .bucket(bucketName)
                                .key(key)
                                .uploadId(uploadId)
                                .partNumber(1).build();

                String etag1 = s3
                                .uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
                                .eTag();

                CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

                UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
                                .uploadId(uploadId)
                                .partNumber(2).build();
                String etag2 = s3
                                .uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
                                .eTag();
                CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

                // Finally call completeMultipartUpload operation to tell S3 to merge
all
                // uploaded
                // parts and finish the multipart operation.
                CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                                .parts(part1, part2)
                                .build();

                CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
                                .bucket(bucketName)
                                .key(key)
                                .uploadId(uploadId)
                                .multipartUpload(completedMultipartUpload)
                                .build();

                s3.completeMultipartUpload(completeMultipartUploadRequest);
```

See the complete example on GitHub.

Delete an object

Build a DeleteObjectRequest and supply a bucket name and key name. Use the S3Client's `deleteObject` method, and pass it the name of a bucket and object to delete. *The specified bucket and object key must exist, or the service will return an error.*

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
                DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                                .bucket(bucketName)
                                .key(key)
```

```
                                .build();

                s3.deleteObject(deleteObjectRequest);
```

See the complete example on GitHub.

Copy an object

Build a CopyObjectRequest and supply a bucket name that the object is coped into, a URL encoded string value (see the URLEncoder.encode method), and the key name of the object. Use the S3Client's `copyObject` method, and pass the CopyObjectRequest object. *The specified bucket and object key must exist, or the service will return an error.*

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <objectKey> <fromBucket> <toBucket>

                Where:
                    objectKey - The name of the object (for example, book.pdf).
                    fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
                    toBucket - The S3 bucket to copy the object to (for example,
bucket2).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String objectKey = args[0];
        String fromBucket = args[1];
        String toBucket = args[2];
        System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        copyBucketObject(s3, fromBucket, objectKey, toBucket);
        s3.close();
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
                .sourceBucket(fromBucket)
                .sourceKey(objectKey)
                .destinationBucket(toBucket)
                .destinationKey(objectKey)
                .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return "";
    }
}
```

See the complete example on GitHub.

List objects

Build a ListObjectsRequest and supply the bucket name. Then invoke the S3Client's `listObjects` method and pass the `ListObjectsRequest` object. This method returns a ListObjectsResponse that contains all of the objects in the bucket. You can invoke this object's *contents* method to get a list of objects. You can iterate through this list to display the objects, as shown in the following code example.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;
```

Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket from which objects are read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                System.out.print("\n The object is " + calKb(myValue.size()) + " KBs");
                System.out.print("\n The owner is " + myValue.owner());
            }
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // convert bytes to kbs.
    private static long calKb(Long val) {
        return val / 1024;
    }
}
```

See the complete example on GitHub.

More examples

The Code examples section of this guide contains more examples of working with Amazon S3 objects including how to download an object.

Work with Amazon S3 pre-signed URLs

Pre-signed URLs provide temporary access to private S3 objects without requiring users to have AWS credentials or permissions.

For example, assume Alice has access to an S3 object, and she wants to temporarily share access to that object with Bob. Alice can generate a pre-signed GET request to share with Bob so that he can download the object without requiring access to Alice's credentials. You can generate pre-signed URLs for HTTP GET and for HTTP PUT requests.

Generate a pre-signed URL for an object, then download it (GET request)

The following example consists of two parts.

- Part 1: Alice generates the pre-signed URL for an object.
- Part 2: Bob downloads the object by using the pre-signed URL.

Part 1: Generate the URL

Alice already has an object in an S3 bucket. She uses the following code to generate a URL string that Bob can use in a subsequent GET request.

Imports

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

```
    /* Create a pre-signed URL to download an object in a subsequent GET request. */
    public String createPresignedGetUrl(String bucketName, String keyName) {
        try (S3Presigner presigner = S3Presigner.create()) {

            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            GetObjectPresignRequest presignRequest = GetObjectPresignRequest.builder()
```

```
                    .signatureDuration(Duration.ofMinutes(10))  // The URL will expire
in 10 minutes.
                    .getObjectRequest(objectRequest)
                    .build();

            PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
            logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
            logger.info("HTTP method: [{}]", presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
```

Part 2: Download the object

Bob uses one of the following three code options to download the object. Alternatively, he could use a browser to perform the GET request.

Use JDK `HttpURLConnection` (since v1.1)

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
    public byte[] useHttpUrlConnectionToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
Capture the response body to a byte array.

        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setRequestMethod("GET");
            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                IoUtils.copy(content, byteArrayOutputStream);
            }
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Use JDK `HttpClient` (since v11)

```
    /* Use the JDK HttpClient (since v11) class to do the download. */
    public byte[] useHttpClientToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
 Capture the response body to a byte array.

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpResponse<InputStream> response = httpClient.send(requestBuilder
                            .uri(presignedUrl.toURI())
                            .GET()
                            .build(),
                    HttpResponse.BodyHandlers.ofInputStream());

            IoUtils.copy(response.body(), byteArrayOutputStream);

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Use `SdkHttpClient` from the SDK for Java

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToPut(String presignedUrlString) {

        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
 Capture the response body to a byte array.
        try {
            URL presignedUrl = new URL(presignedUrlString);
            SdkHttpRequest request = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.GET)
                    .uri(presignedUrl.toURI())
                    .build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
```

```
                .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
 sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                    abortableInputStream -> {
                        try {
                            IoUtils.copy(abortableInputStream,
 byteArrayOutputStream);
                        } catch (IOException e) {
                            throw new RuntimeException(e);
                        }
                    },
                    () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
 response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

See the complete example and test on GitHub.

Generate a pre-signed URL for an upload, then upload a file (PUT request)

The following example consists of two parts.

- Part 1: Alice generates the pre-signed URL to upload an object.
- Part 2: Bob uploads a file by using the pre-signed URL.

Part 1: Generate the URL

Alice already has an S3 bucket. She uses the following code to generate a URL string that Bob can use in a subsequent PUT request.

Imports

```
import com.example.s3.util.PresignUrlUtils;
```

```
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

```
    /* Create a presigned URL to use in a subsequent PUT request */
    public String createPresignedUrl(String bucketName, String keyName, Map<String,
 String> metadata) {
        try (S3Presigner presigner = S3Presigner.create()) {

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .metadata(metadata)
                    .build();
```

```
            PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))  // The URL expires in
10 minutes.
                    .putObjectRequest(objectRequest)
                    .build();


            PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
            String myURL = presignedRequest.url().toString();
            logger.info("Presigned URL to upload a file to: [{}]", myURL);
            logger.info("HTTP method: [{}]", presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
```

Part 2: Upload a file object

Bob uses one of the following three code options to upload a file.

Use JDK `HttpURLConnection` (since v1.1)

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
    public void useHttpUrlConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" + k,
v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                 FileChannel inChannel = file.getChannel()) {
                ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

                while (inChannel.read(buffer) > 0) {
                    buffer.flip();
```

```
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
                buffer.clear();
            }
        } catch (IOException e) {
            logger.error(e.getMessage(), e);
        }

        out.close();
        connection.getResponseCode();
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

Use JDK `HttpClient` (since v11)

```
    /* Use the JDK HttpClient (since v11) class to do the upload. */
    public void useHttpClientToPut(String presignedUrlString, File fileToPut,
 Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            final HttpResponse<Void> response = httpClient.send(requestBuilder
                            .uri(new URL(presignedUrlString).toURI())

 .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
                            .build(),
                    HttpResponse.BodyHandlers.discarding());

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Use SdkHttpClient from the SDK for Java

```
    /* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
    public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
 Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        try {
            URL presignedUrl = new URL(presignedUrlString);

            SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.PUT)
                    .uri(presignedUrl.toURI());
            // Add headers
            metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k, v));
            // Finish building the request.
            SdkHttpRequest request = requestBuilder.build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
 FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
 sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}", response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

See the complete example and test on GitHub.

Cross-Region access for Amazon S3

When you work with Amazon Simple Storage Service (Amazon S3) buckets, you usually know the AWS Region for the bucket. The Region you work with is determined when you create the S3 client.

However, sometimes you might need to work with a specific bucket, but you don't know if it's located in the same Region that's set for the S3 client.

Instead of making more calls to determine the bucket Region, you can use the SDK to enable access to S3 buckets across different Regions.

Setup

Support for cross-Region access became available with version `2.20.111` of the SDK. Use this version or a later one in your Maven build file for the `s3` dependency as shown in the following snippet.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.111</version>
</dependency>
```

Next, when you create your S3 client, enable cross-Region access as shown in the snippet. By default, access is not enabled.

```
S3AsyncClient client = S3AsyncClient.builder()
                                    .crossRegionAccessEnabled(true)
                                    .build();
```

How the SDK provides cross-Region access

When you reference an existing bucket in a request, such as when you use the `putObject` method, the SDK initiates a request to the Region configured for the client.

If the bucket does not exist in that specific Region, the error response includes the actual Region where the bucket resides. The SDK then uses the correct Region in a second request.

To optimize future requests to the same bucket, the SDK caches this Region mapping in the client.

Considerations

When you enable cross-Region bucket access, be aware that the first API call might result in increased latency if the bucket isn't in the client's configured Region. However, subsequent calls benefit from cached Region information, resulting in improved performance.

When you enable cross-Region access, access to the bucket is not affected. The user must be authorized to access the bucket in whatever Region it resides.

Amazon S3 checksums with AWS SDK for Java

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

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

Amazon S3 currently supports four checksum algorithms: SHA-1, SHA-256, CRC-32, and CRC-32C. You have the flexibility to choose the algorithm that best fits your needs and let the SDK calculate the checksum. Alternatively, you can specify their own pre-computed checksum value by using one of the four supported algorithms.

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

Upload an object

You upload objects to Amazon S3 by using the [putObject](#) method of the `S3Client`. Use the `checksumAlgorithm` method of the builder for the `PutObjectRequest` to enable checksum computation and specify the algorithm. Valid values for the algorithm are `CRC32`, `CRC32C`, `SHA1`, and `SHA256`.

The following code snippet shows a request to upload an object with a CRC-32 checksum. When the SDK sends the request, it calculates the CRC-32 checksum and uploads the object. Amazon S3 stores the checksum with the object.

```
    public void putObjectWithChecksum() {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.CRC32),
                RequestBody.fromString("This is a test"));
    }
```

If the checksum that the SDK calculates doesn't match the checksum that Amazon S3 calculates when it receives the request, an error is returned.

Use a pre-calculated checksum value

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

The following example shows a request with a pre-calculated SHA-256 checksum.

```
    public void putObjectWithPrecalculatedChecksum(String filePath) {
        String checksum = calculateChecksum(filePath, "SHA-256");

        s3Client.putObject((b -> b
                .bucket(bucketName)
                .key(key)
                .checksumSHA256(checksum)),
                RequestBody.fromFile(Paths.get(filePath)));
    }
```

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

Multipart uploads

You can also use checksums with multipart uploads. The SDK for Java 2.x provides two options to use checksums with multipart uploads. The first option uses the `S3TransferManager`.

The following transfer manager example specifies the SHA1 algorithm for the upload.

```
    public void multipartUploadWithChecksumTm(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b
                        .bucket(bucketName)
                        .key(key)
                        .checksumAlgorithm(ChecksumAlgorithm.SHA1))
                .source(Paths.get(filePath))
                .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

The second option uses the [`S3Client` API] (or the [`S3AsyncClient` API]) to perform the multipart upload. If you specify a checksum with this approach, you must specify the algorithm to use on the initiation of the upload. You must also specify the algorithm for each part request and provide the checksum calculated for each part after it is uploaded.

```
    public void multipartUploadWithChecksumS3Client(String filePath) {
        ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
 s3Client.createMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .checksumAlgorithm(algorithm) // Checksum specified on each
 part.
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
```

```
                        .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

Code for the complete examples and tests are in the GitHub code examples repository.

Download an object

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

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

```
    public GetObjectResponse getObjectWithChecksum() {
        return s3Client.getObject(b -> b
                        .bucket(bucketName)
                        .key(key)
                        .checksumMode(ChecksumMode.ENABLED))
                .response();
    }
```

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

An object in Amazon S3 can have multiple checksums, but only one checksum is validated on download. The following precedence— based on the efficiency of the checksum algorithm—determines which checksum the SDK validates:

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

For example, if a response contains both CRC-32 and SHA-256 checksums, only the CRC-32 checksum is validated.

Use a performant S3 client: AWS CRT-based S3 client

The AWS CRT-based S3 client—built on top of the AWS Common Runtime (CRT)—is an alternative S3 asynchronous client. It transfers objects to and from Amazon Simple Storage Service (Amazon S3) with enhanced performance and reliability by automatically using Amazon S3's multipart upload API and byte-range fetches.

The AWS CRT-based S3 client improves transfer reliability in case there is a network failure. Reliability is improved by retrying individual failed parts of a file transfer without restarting the transfer from the beginning.

In addition, the AWS CRT-based S3 client offers enhanced connection pooling and Domain Name System (DNS) load balancing, which also improves throughput.

You can use the AWS CRT-based S3 client in place of the SDK's standard S3 asynchronous client and take advantage of its improved throughput right away.

AWS CRT-based components in the SDK

The AWS CRT-based *S3* client, described in this topic, and the AWS CRT-based *HTTP* client are different components in the SDK.

The **AWS CRT-based S3 client** is an implementation of the S3AsyncClient interface and is used for working with the Amazon S3 service. It is an alternative to the Java-based implementation of the `S3AsyncClient` interface and offers several benefits.

The AWS CRT-based HTTP client is an implementation of the SdkAsyncHttpClient interface and is used for general HTTP communication. It is an alternative to the Netty implementation of the `SdkAsyncHttpClient` interface and offers several advantages.

Although both components use libraries from the AWS Common Runtime, the AWS CRT-based S3 client uses the aws-c-s3 library and supports the S3 multipart upload API features. Since the AWS CRT-based HTTP client is meant for general purpose use, it does not support the S3 multipart upload API features.

Add dependencies to use the AWS CRT-based S3 client

To use the AWS CRT-based S3 client, add the following two dependencies to your Maven project file. The example shows the minimum versions to use. Search the Maven central repository for the most recent versions of the s3 and aws-crt artifacts.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.68</version>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
  <artifactId>aws-crt</artifactId>
  <version>0.21.16</version>
</dependency>
```

Create an instance of the AWS CRT-based S3 client

Create an instance of the AWS CRT-based S3 client with default settings as shown in the following code snippet.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
```

To configure the client, use the AWS CRT client builder. You can switch from the standard S3 asynchronous client to AWS CRT-based client by changing the builder method.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
```

```
S3AsyncClient s3AsyncClient =
        S3AsyncClient.crtBuilder()
                     .credentialsProvider(DefaultCredentialsProvider.create())
                     .region(Region.US_WEST_2)
                     .targetThroughputInGbps(20.0)
                     .minimumPartSizeInBytes(8 * 1025 * 1024L)
                     .build();
```

Note

Some of the settings in the standard builder might not be currently supported in the AWS CRT client builder. Get the standard builder by calling `S3AsyncClient#builder()`.

Use the AWS CRT-based S3 client

Use the AWS CRT-based S3 client to call Amazon S3 API operations. The following example demonstrates the PutObject and GetObject operations available through the AWS SDK for Java.

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;


S3AsyncClient s3Client = S3AsyncClient.crtCreate();

// Upload a local file to Amazon S3.
PutObjectResponse putObjectResponse =
      s3Client.putObject(req -> req.bucket(<BUCKET_NAME>)
                                   .key(<KEY_NAME>),
                        AsyncRequestBody.fromFile(Paths.get(<FILE_NAME>)))
              .join();

// Download an object from Amazon S3 to a local file.
GetObjectResponse getObjectResponse =
      s3Client.getObject(req -> req.bucket(<BUCKET_NAME>)
                                   .key(<KEY_NAME>),
                        AsyncResponseTransformer.toFile(Paths.get(<FILE_NAME>)))
              .join();
```

Transfer files and directories with the Amazon S3 Transfer Manager

The Amazon S3 Transfer Manager is an open source, high level file transfer utility for the AWS SDK for Java 2.x. Use it to transfer files and directories to and from Amazon Simple Storage Service (Amazon S3).

When built on top of the AWS CRT-based S3 client, the S3 Transfer Manager can take advantage of performance improvements such as the multipart upload API and byte-range fetches.

With the S3 Transfer Manager, you can also monitor a transfer's progress in real time and pause the transfer for later execution.

Get started

Add dependencies to your build file

To use the S3 Transfer Manager with enhanced performance based on the AWS CRT-based S3 client, configure your build file with the following dependencies.

- Use version *2.19.1* or higher of the SDK for Java 2.x.
- Add the `s3-transfer-manager` artifact as a dependency.
- Add the `aws-crt` artifact as a dependency at version *0.20.3* or higher.

The following code example shows how to configure your project dependencies for Maven.

```
<project>
    <properties>
      <aws.sdk.version>2.19.1</aws.sdk.version>
   </properties>
   <dependencyManagement>
    <dependencies>
        <dependency>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>bom</artifactId>
          <version>${aws.sdk.version}</version>
          <type>pom</type>
          <scope>import</scope>
        </dependency>
    </dependencies>
   </dependencyManagement>
   <dependencies>
```

```
    <dependency>
       <groupId>software.amazon.awssdk</groupId>
       <artifactId>s3-transfer-manager</artifactId>
    </dependency>
    <dependency>
       <groupId>software.amazon.awssdk.crt</groupId>
       <artifactId>aws-crt</artifactId>
       <version>0.20.3</version>
    </dependency>
   </dependencies>
</project>
```

Search the Maven central repository for the most recent versions of the s3-transfer-manager and aws-crt artifacts.

Create an instance of the S3 Transfer Manager

The following snippet shows how to create a S3TransferManager instance with default settings.

```
        S3TransferManager transferManager = S3TransferManager.create();
```

The following example shows how to configure a S3 Transfer Manager with custom settings. In this example, a AWS CRT-based S3AsyncClient instance is used as the underlying client for the S3 Transfer Manager.

```
        S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .targetThroughputInGbps(20.0)
                .minimumPartSizeInBytes(8 * MB)
                .build();

        S3TransferManager transferManager = S3TransferManager.builder()
                .s3Client(s3AsyncClient)
                .build();
```

Note

If the `aws-crt` dependency is not included in the build file, the S3 Transfer Manager is built on top of the standard S3 asynchronous client used in the SDK for Java 2.x.

Upload a file to an S3 bucket

The following example shows a file upload example along with the optional use of a LoggingTransferListener, which logs the progress of the upload.

To upload a file to Amazon S3 using the S3 Transfer Manager, pass an `UploadFileRequest` object to the `S3TransferManager`'s uploadFile method.

The FileUpload object returned from the `uploadFile` method represents the upload process. After the request finishes, the CompletedFileUpload object contains information about the upload.

```
    public String uploadFile(S3TransferManager transferManager, String bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Imports

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

Download a file from an S3 bucket

The following example shows a download example along with the optional use of a LoggingTransferListener, which logs the progress of the download.

To download an object from an S3 bucket using the S3 Transfer Manager, build a DownloadFileRequest object and pass it to the downloadFile method.

The FileDownload object returned by the `S3TransferManager`'s `downloadFile` method represents the file transfer. After the download completes, the CompletedFileDownload contains access to information about the download.

```
    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile = transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult = downloadFile.completionFuture().join();
        logger.info("Content length [{}]", downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Imports

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
```

```
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

Copy an Amazon S3 object to another bucket

The following example shows how to copy an object with the S3 Transfer Manager.

To begin the copy of an object from an S3 bucket to another bucket, create a basic CopyObjectRequest instance.

Next, wrap the basic `CopyObjectRequest` in a CopyRequest that can be used by the S3 Transfer Manager.

The `Copy` object returned by the `S3TransferManager`'s `copy` method represents the copy process. After the copy process completes, the CompletedCopy object contains details about the response.

```
    public String copyObject(S3TransferManager transferManager, String bucketName,
            String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
                .sourceBucket(bucketName)
                .sourceKey(key)
                .destinationBucket(destinationBucket)
                .destinationKey(destinationKey)
                .build();

        CopyRequest copyRequest = CopyRequest.builder()
                .copyObjectRequest(copyObjectRequest)
                .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

Note

To perform a cross-Region copy with the S3 Transfer Manager, enable `crossRegionAccessEnabled` on the AWS CRT-based S3 client builder as shown in the following snippet.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
                .crossRegionAccessEnabled(true)
                .build();

S3TransferManager transferManager = S3TransferManager.builder()
                .s3Client(s3AsyncClient)
                .build();
```

Imports

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;
```

Upload a local directory to an S3 bucket

The following example demonstrates how you can upload a local directory to S3.

Start by calling the uploadDirectory method of the `S3TransferManager` instance, passing in an UploadDirectoryRequest.

The DirectoryUpload object represents the upload process, which generates a CompletedDirectoryUpload when the request completes. The `CompleteDirectoryUpload` object contains information about the results of the transfer, including which files failed to transfer.

```
    public Integer uploadDirectory(S3TransferManager transferManager,
            URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
 transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());
```

```
        CompletedDirectoryUpload completedDirectoryUpload =
 directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
 fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

Imports

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

Download S3 bucket objects to a local directory

You can download the objects in an S3 bucket to a local directory as shown in the following example.

To download the objects in an S3 bucket to a local directory, begin by calling the downloadDirectory method of the Transfer Manager, passing in a DownloadDirectoryRequest.

The DirectoryDownload object represents the download process, which generates a CompletedDirectoryDownload when the request completes. The `CompleteDirectoryDownload` object contains information about the results of the transfer, including which files failed to transfer.

```
    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
```

```
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

Imports

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;
```

See complete examples

GitHub contains the complete code for all examples on this page.

Work with Amazon Simple Notification Service

With Amazon Simple Notification Service, you can easily push real-time notification messages from your applications to subscribers over multiple communication channels. This topic describes how to perform some of the basic functions of Amazon SNS.

Create a topic

A **topic** is a logical grouping of communication channels that defines which systems to send a message to, for example, fanning out a message to AWS Lambda and an HTTP webhook. You send messages to Amazon SNS, then they're distributed to the channels defined in the topic. This makes the messages available to subscribers.

To create a topic, first build a CreateTopicRequest object, with the name of the topic set using the `name()` method in the builder. Then, send the request object to Amazon SNS by using the `createTopic()` method of the SnsClient. You can capture the result of this request as a CreateTopicResponse object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
    public static String createSNSTopic(SnsClient snsClient, String topicName ) {

        CreateTopicResponse result = null;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();
        } catch (SnsException e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

See the complete example on GitHub.

List your Amazon SNS topics

To retrieve a list of your existing Amazon SNS topics, build a ListTopicsRequest object. Then, send the request object to Amazon SNS by using the `listTopics()` method of the `SnsClient`. You can capture the result of this request as a ListTopicsResponse object.

The following code snippet prints out the HTTP status code of the request and a list of Amazon Resource Names (ARNs) for your Amazon SNS topics.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
    public static void listSNSTopics(SnsClient snsClient) {

        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                    .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode() +
 "\n\nTopics\n\n" + result.topics());

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

See the complete example on GitHub.

Subscribe an endpoint to a topic

After you create a topic, you can configure which communication channels will be endpoints for that topic. Messages are distributed to these endpoints after Amazon SNS receives them.

To configure a communication channel as an endpoint for a topic, subscribe that endpoint to the topic. To start, build a SubscribeRequest object. Specify the communication channel (for example, `lambda` or `email`) as the `protocol()`. Set the `endpoint()` to the relevant output location (for example, the ARN of a Lambda function or an email address), and then set the ARN of the topic to which you want to subscribe as the `topicArn()`. Send the request object to Amazon SNS by using the `subscribe()` method of the `SnsClient`. You can capture the result of this request as a SubscribeResponse object.

The following code snippet shows how to subscribe an email address to a topic.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
```

Code

```
    public static void subEmail(SnsClient snsClient, String topicArn, String email) {

        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("email")
                .endpoint(email)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n
 Status is " + result.sdkHttpResponse().statusCode());

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

See the complete example on GitHub.

Publish a message to a topic

After you have a topic and one or more endpoints configured for it, you can publish a message to it. To start, build a PublishRequest object. Specify the `message()` to send, and the ARN of the topic (`topicArn()`) to send it to. Then, send the request object to Amazon SNS by using the `publish()` method of the `SnsClient`. You can capture the result of this request as a PublishResponse object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
    public static void pubTopic(SnsClient snsClient, String message, String topicArn) {

        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out.println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

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

See the complete example on GitHub.

Unsubscribe an endpoint from a topic

You can remove the communication channels configured as endpoints for a topic. After doing that, the topic itself continues to exist and distribute messages to any other endpoints configured for that topic.

To remove a communication channel as an endpoint for a topic, unsubscribe that endpoint from the topic. To start, build an UnsubscribeRequest object and set the ARN of the topic you want to unsubscribe from as the `subscriptionArn()`. Then send the request object to SNS by using the `unsubscribe()` method of the `SnsClient`. You can capture the result of this request as an UnsubscribeResponse object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
```

Code

```
    public static void unSub(SnsClient snsClient, String subscriptionArn) {

        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);

            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                + "\n\nSubscription was removed for " + request.subscriptionArn());

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

See the complete example on GitHub.

Delete a topic

To delete an Amazon SNS topic, first build a DeleteTopicRequest object with the ARN of the topic set as the `topicArn()` method in the builder. Then send the request object to Amazon SNS by using the `deleteTopic()` method of the `SnsClient`. You can capture the result of this request as a DeleteTopicResponse object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn ) {

        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

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

See the complete example on GitHub.

For more information, see the Amazon Simple Notification Service Developer Guide.

Work with Amazon Simple Queue Service

This section provides examples of programming Amazon Simple Queue Service using the AWS SDK for Java 2.x.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- Work with Amazon Simple Queue Service message queues
- Send, receive, and delete Amazon Simple Queue Service messages

Work with Amazon Simple Queue Service message queues

A *message queue* is the logical container used for sending messages reliably in Amazon Simple Queue Service. There are two types of queues: *standard* and *first-in, first-out* (FIFO). To learn more about queues and the differences between these types, see the Amazon Simple Queue Service Developer Guide.

This topic describes how to create, list, delete, and get the URL of an Amazon Simple Queue Service queue by using the AWS SDK for Java.

The `sqsClient` variable that is used in the following examples can be created from the following snippet.

```
SqsClient sqsClient = SqsClient.create();
```

When you create an `SqsClient` by using the static `create()` method, the SDK configures the Region by using the default region provider chain and the credentials by using the default credentials provider chain.

Create a queue

Use the `SqsClient’s` `createQueue` method, and provide a `CreateQueueRequest` object that describes the queue parameters as shown in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();

            sqsClient.createQueue(createQueueRequest);
```

See the complete sample on GitHub.

List queues

To list the Amazon Simple Queue Service queues for your account, call the `SqsClient’s` `listQueues` method with a `ListQueuesRequest` object.

When you use the form of the `listQueues` method that takes no parameters, the service returns *all queues*—up to 1,000 queues.

You can supply a queue name prefix to the `ListQueuesRequest` object to limit the results to queues that match that prefix as shown in the following code.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
 ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
 sqsClient.listQueues(listQueuesRequest);
```

```
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

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

See the complete sample on GitHub.

Get the URL for a queue

The following code shows how to get the URL for a queue by calling the `SqsClient’s` `getQueueUrl` method with a `GetQueueUrlRequest` object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
            GetQueueUrlResponse getQueueUrlResponse =
 
 sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            String queueUrl = getQueueUrlResponse.queueUrl();
            return queueUrl;
```

See the complete sample on GitHub.

Delete a queue

Provide the queue’s URL to the `DeleteQueueRequest` object. Then call the `SqsClient’s` `deleteQueue` method to delete a queue as shown in the following code.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
```

```
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {

        try {

            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();

            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);

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

See the complete sample on GitHub.

More information

- CreateQueue in the Amazon Simple Queue Service API Reference
- GetQueueUrl in the Amazon Simple Queue Service API Reference
- ListQueues in the Amazon Simple Queue Service API Reference
- DeleteQueue in the Amazon Simple Queue Service API Reference

Send, receive, and delete Amazon Simple Queue Service messages

A message is a piece of data that can be sent and received by distributed components. Messages are always delivered using an SQS Queue.

The `sqsClient` variable that is used in the following examples can be created from the following snippet.

```
SqsClient sqsClient = SqsClient.create();
```

When you create an `SqsClient` by using the static `create()` method, the SDK configures the Region by using the default region provider chain and the credentials by using the default credentials provider chain.

Send a message

Add a single message to an Amazon Simple Queue Service queue by calling the SqsClient client `sendMessage` method. Provide a SendMessageRequest object that contains the queue's URL, message body, and optional delay value (in seconds).

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
            sqsClient.sendMessage(SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody("Hello world!")
                .delaySeconds(10)
                .build());

            sqsClient.sendMessage(sendMsgRequest);
```

Send multiple messages in a request

Send more than one message in a single request by using the SqsClient `sendMessageBatch` method. This method takes a SendMessageBatchRequest that contains the queue URL and a list of messages to send. (Each message is a SendMessageBatchRequestEntry.) You can also delay sending a specific message by setting a delay value on the message.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
            SendMessageBatchRequest sendMessageBatchRequest =
 SendMessageBatchRequest.builder()
                .queueUrl(queueUrl)

 .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
 1").build(),

 SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
 2").delaySeconds(10).build())
                .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

See the complete sample on GitHub.

Retrieve Messages

Retrieve any messages that are currently in the queue by calling the SqsClient `receiveMessage` method. This method takes a ReceiveMessageRequest that contains the queue URL. You can also specify the maximum number of messages to return. Messages are returned as a list of Message objects.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
        try {
            ReceiveMessageRequest receiveMessageRequest =
 ReceiveMessageRequest.builder()
```

```
                .queueUrl(queueUrl)
                .maxNumberOfMessages(5)
                .build();
            List<Message> messages =
 sqsClient.receiveMessage(receiveMessageRequest).messages();
            return messages;
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
```

See the complete sample on GitHub.

Delete a message after receipt

After receiving a message and processing its contents, delete the message from the queue by sending the message's receipt handle and queue URL to the `SqsClient`'s `deleteMessage` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
 DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
```

See the complete sample on GitHub.

More Info

- How Amazon Simple Queue Service Queues Work in the Amazon Simple Queue Service Developer Guide
- SendMessage in the Amazon Simple Queue Service API Reference
- SendMessageBatch in the Amazon Simple Queue Service API Reference
- ReceiveMessage in the Amazon Simple Queue Service API Reference
- DeleteMessage in the Amazon Simple Queue Service API Reference

Work with Amazon Transcribe

The following example shows how bidirectional streaming works using Amazon Transcribe. Bidirectional streaming implies that there's both a stream of data going to the service and being received back in real time. The example uses Amazon Transcribe streaming transcription to send an audio stream and receive a stream of transcribed text back in real time.

See Streaming Transcription in the Amazon Transcribe Developer Guide to learn more about this feature.

See Getting Started in the Amazon Transcribe Developer Guide to get started using Amazon Transcribe.

Set up the microphone

This code uses the javax.sound.sampled package to stream audio from an input device.

Code

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;

public class Microphone {

    public static TargetDataLine get() throws Exception {
        AudioFormat format = new AudioFormat(16000, 16, 1, true, false);
        DataLine.Info datalineInfo = new DataLine.Info(TargetDataLine.class, format);

        TargetDataLine dataLine = (TargetDataLine) AudioSystem.getLine(datalineInfo);
```

```
        dataLine.open(format);

        return dataLine;
    }
}
```

See the complete example on GitHub.

Create a publisher

This code implements a publisher that publishes audio data from the Amazon Transcribe audio stream.

Code

```
package com.amazonaws.transcribe;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.atomic.AtomicLong;
import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.transcribestreaming.model.AudioEvent;
import software.amazon.awssdk.services.transcribestreaming.model.AudioStream;
import
 software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException;


public class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;

    public AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {
```

```
        s.onSubscribe(new SubscriptionImpl(s, inputStream));
    }

    private class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;

        private SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
 inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
 positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent = audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (TranscribeStreamingException e) {
                    subscriber.onError(e);
                }
            });
        }
```

```
        @Override
        public void cancel() {

        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

See the complete example on GitHub.

Create the client and start the stream

In the main method, create a request object, start the audio input stream and instantiate the publisher with the audio input.

You must also create a StartStreamTranscriptionResponseHandler to specify how to handle the response from Amazon Transcribe.

Then, use the TranscribeStreamingAsyncClient's `startStreamTranscription` method to start the bidirectional streaming.

Imports

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;
import javax.sound.sampled.AudioInputStream;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.transcribestreaming.TranscribeStreamingAsyncClient;
import
 software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException ;
import
 software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionRequest;
import software.amazon.awssdk.services.transcribestreaming.model.MediaEncoding;
import software.amazon.awssdk.services.transcribestreaming.model.LanguageCode;
import
 software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionResponseHand
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptEvent;
```

Code

```
    public static void convertAudio(TranscribeStreamingAsyncClient client) throws
 Exception {

        try {

            StartStreamTranscriptionRequest request =
 StartStreamTranscriptionRequest.builder()
                    .mediaEncoding(MediaEncoding.PCM)
                    .languageCode(LanguageCode.EN_US)
                    .mediaSampleRateHertz(16_000).build();

            TargetDataLine mic = Microphone.get();
            mic.start();

            AudioStreamPublisher publisher = new AudioStreamPublisher(new
 AudioInputStream(mic));

            StartStreamTranscriptionResponseHandler response =
```

```
                    StartStreamTranscriptionResponseHandler.builder().subscriber(e -> {
                        TranscriptEvent event = (TranscriptEvent) e;
                        event.transcript().results().forEach(r ->
r.alternatives().forEach(a -> System.out.println(a.transcript())));
                    }).build();

            // Keeps Streaming until you end the Java program
            client.startStreamTranscription(request, publisher, response);

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

See the complete example on GitHub.

More information

- How It Works in the Amazon Transcribe Developer Guide.
- Getting Started With Streaming Audio in the Amazon Transcribe Developer Guide.

SDK for Java 2.x code examples

The code examples in this topic show you how to use the AWS SDK for Java 2.x with AWS.

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

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

Cross-service examples are sample applications that work across multiple AWS services.

Examples

Actions and scenarios using SDK for Java 2.x

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS services.

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

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

Services

- Amazon Bedrock examples using SDK for Java 2.x
- Amazon Bedrock Runtime examples using SDK for Java 2.x
- CloudFront examples using SDK for Java 2.x
- CloudWatch examples using SDK for Java 2.x
- CloudWatch Events examples using SDK for Java 2.x
- CloudWatch Logs examples using SDK for Java 2.x
- Amazon Cognito Identity examples using SDK for Java 2.x
- Amazon Cognito Identity Provider examples using SDK for Java 2.x
- Amazon Comprehend examples using SDK for Java 2.x
- DynamoDB examples using SDK for Java 2.x
- Amazon EC2 examples using SDK for Java 2.x
- Amazon ECR examples using SDK for Java 2.x
- Amazon ECS examples using SDK for Java 2.x
- Elastic Load Balancing - Version 2 examples using SDK for Java 2.x
- MediaStore examples using SDK for Java 2.x
- OpenSearch Service examples using SDK for Java 2.x
- EventBridge examples using SDK for Java 2.x
- Forecast examples using SDK for Java 2.x
- AWS Glue examples using SDK for Java 2.x
- HealthImaging examples using SDK for Java 2.x
- IAM examples using SDK for Java 2.x
- AWS IoT examples using SDK for Java 2.x
- AWS IoT data examples using SDK for Java 2.x
- Amazon Keyspaces examples using SDK for Java 2.x
- Kinesis examples using SDK for Java 2.x
- AWS KMS examples using SDK for Java 2.x
- Lambda examples using SDK for Java 2.x
- MediaConvert examples using SDK for Java 2.x
- Migration Hub examples using SDK for Java 2.x

- Amazon Personalize examples using SDK for Java 2.x
- Amazon Personalize Events examples using SDK for Java 2.x
- Amazon Personalize Runtime examples using SDK for Java 2.x
- Amazon Pinpoint examples using SDK for Java 2.x
- Amazon Pinpoint SMS and Voice API examples using SDK for Java 2.x
- Amazon Polly examples using SDK for Java 2.x
- Amazon RDS examples using SDK for Java 2.x
- Amazon Redshift examples using SDK for Java 2.x
- Amazon Rekognition examples using SDK for Java 2.x
- Route 53 domain registration examples using SDK for Java 2.x
- Amazon S3 examples using SDK for Java 2.x
- S3 Glacier examples using SDK for Java 2.x
- SageMaker examples using SDK for Java 2.x
- Secrets Manager examples using SDK for Java 2.x
- Amazon SES examples using SDK for Java 2.x
- Amazon SES API v2 examples using SDK for Java 2.x
- Amazon SNS examples using SDK for Java 2.x
- Amazon SQS examples using SDK for Java 2.x
- Step Functions examples using SDK for Java 2.x
- AWS STS examples using SDK for Java 2.x
- AWS Support examples using SDK for Java 2.x
- Systems Manager examples using SDK for Java 2.x
- Amazon Textract examples using SDK for Java 2.x
- Amazon Transcribe examples using SDK for Java 2.x

API Gateway examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with API Gateway.

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

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

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

`CreateDeployment`

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

SDK for Java 2.x

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

```
    public static String createNewDeployment(ApiGatewayClient apiGateway, String restApiId, String stageName) {

        try {
            CreateDeploymentRequest request = CreateDeploymentRequest.builder()
                    .restApiId(restApiId)
                    .description("Created using the AWS API Gateway Java API")
                    .stageName(stageName)
                    .build();

            CreateDeploymentResponse response = apiGateway.createDeployment(request);
            System.out.println("The id of the deployment is " + response.id());
            return response.id();
```

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

- For API details, see CreateDeployment in *AWS SDK for Java 2.x API Reference*.

CreateRestApi

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

SDK for Java 2.x

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

```
    public static String createAPI(ApiGatewayClient apiGateway, String restApiId,
 String restApiName) {

        try {
            CreateRestApiRequest request = CreateRestApiRequest.builder()
                    .cloneFrom(restApiId)
                    .description("Created using the Gateway Java API")
                    .name(restApiName)
                    .build();

            CreateRestApiResponse response = apiGateway.createRestApi(request);
            System.out.println("The id of the new api is " + response.id());
            return response.id();

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

```
    }
```

- For API details, see CreateRestApi in *AWS SDK for Java 2.x API Reference*.

DeleteDeployment

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

SDK for Java 2.x

Note

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

```
    public static void deleteSpecificDeployment(ApiGatewayClient apiGateway, String
 restApiId, String deploymentId) {

        try {
            DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()
                    .restApiId(restApiId)
                    .deploymentId(deploymentId)
                    .build();

            apiGateway.deleteDeployment(request);
            System.out.println("Deployment was deleted");

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

- For API details, see DeleteDeployment in *AWS SDK for Java 2.x API Reference*.

DeleteRestApi

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

SDK for Java 2.x

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

```
    public static void deleteAPI(ApiGatewayClient apiGateway, String restApiId) {

        try {
            DeleteRestApiRequest request = DeleteRestApiRequest.builder()
                    .restApiId(restApiId)
                    .build();

            apiGateway.deleteRestApi(request);
            System.out.println("The API was successfully deleted");

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

- For API details, see DeleteRestApi in *AWS SDK for Java 2.x API Reference*.

Application Auto Scaling examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Application Auto Scaling.

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

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

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

Actions

DeleteScalingPolicy

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExceptio
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetReques
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
```

```
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DisableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).\s
               policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
 ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
 ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        String tableId = args[0];
        String policyName = args[1];

        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
    }
```

```
    public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
        try {
            DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
                .policyName(policyName)
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deleteScalingPolicy(delSPRequest);
            System.out.println(policyName +" was deleted successfully.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the scaling policy was deleted
    public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceId(tableId)
            .build();

        DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    public static void deregisterScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        try {
            DeregisterScalableTargetRequest targetRequest =
DeregisterScalableTargetRequest.builder()
                .scalableDimension(tableWCUs)
```

```
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deregisterScalableTarget(targetRequest);
            System.out.println("The scalable target was deregistered.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    public static void verifyTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
 DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
 appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }
}
```

- For API details, see DeleteScalingPolicy in *AWS SDK for Java 2.x API Reference*.

RegisterScalableTarget

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExceptio
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRespons
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecification;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetRequest;
import software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicyCon
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
```

```
                <tableId> <roleARN> <policyName>\s

            Where:
                tableId - The table Id value (for example, table/Music).
                roleARN - The ARN of the role that has ApplicationAutoScaling
 permissions.
                policyName - The name of the policy to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        System.out.println("This example registers an Amazon DynamoDB table, which
 is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
 ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
 ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
 tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
 policyName);
    }

    public static void registerScalableTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
 ScalableDimension tableWCUs) {
        try {
            RegisterScalableTargetRequest targetRequest =
 RegisterScalableTargetRequest.builder()
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
```

```
                .roleARN(roleARN)
                .minCapacity(5)
                .maxCapacity(10)
                .build();

            appAutoScalingClient.registerScalableTarget(targetRequest);
            System.out.println("You have registered " + tableId);

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the target was created.
    public static void verifyTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
 DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
 appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    // Configure a scaling policy.
    public static void configureScalingPolicy(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs, String policyName) {
        // Check if the policy exists before creating a new one.
        DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
 appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
            .serviceNamespace(ns)
            .resourceId(tableId)
            .scalableDimension(tableWCUs)
            .build());

        if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
```

```
            // If policies exist, consider updating an existing policy instead of
creating a new one.
            System.out.println("Policy already exists. Consider updating it
instead.");
            List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
            for (ScalingPolicy pol : polList) {
                System.out.println("Policy name:" +pol.policyName());
            }
        } else {
            // If no policies exist, proceed with creating a new policy.
            PredefinedMetricSpecification specification =
PredefinedMetricSpecification.builder()

.predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
                .build();

            TargetTrackingScalingPolicyConfiguration policyConfiguration =
TargetTrackingScalingPolicyConfiguration.builder()
                .predefinedMetricSpecification(specification)
                .targetValue(50.0)
                .scaleInCooldown(60)
                .scaleOutCooldown(60)
                .build();

            PutScalingPolicyRequest putScalingPolicyRequest =
PutScalingPolicyRequest.builder()
                .targetTrackingScalingPolicyConfiguration(policyConfiguration)
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .policyName(policyName)
                .policyType(PolicyType.TARGET_TRACKING_SCALING)
                .build();

            try {
                appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
                System.out.println("You have successfully created a scaling policy
for an Application Auto Scaling scalable target");
            } catch (ApplicationAutoScalingException e) {
                System.err.println("Error: " + e.awsErrorDetails().errorMessage());
            }
        }
    }
```

```
}
```

- For API details, see RegisterScalableTarget in *AWS SDK for Java 2.x API Reference*.

Application Recovery Controller examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Application Recovery Controller.

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

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

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

`GetRoutingControlState`

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

SDK for Java 2.x

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

```
    public static GetRoutingControlStateResponse
getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
```

```
            String routingControlArn) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region())).build();
                return client.getRoutingControlState(
                        GetRoutingControlStateRequest.builder()
                                .routingControlArn(routingControlArn).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- For API details, see GetRoutingControlState in *AWS SDK for Java 2.x API Reference*.

`UpdateRoutingControlState`

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

SDK for Java 2.x

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

```
    public static UpdateRoutingControlStateResponse
 updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
```

```
            String routingControlArn,
            String routingControlState) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region()))
                        .build();
                return client.updateRoutingControlState(
                        UpdateRoutingControlStateRequest.builder()

 .routingControlArn(routingControlArn).routingControlState(routingControlState).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- For API details, see UpdateRoutingControlState in *AWS SDK for Java 2.x API Reference*.

Aurora examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Aurora.

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

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

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

Get started

Hello Aurora

The following code examples show how to get started using Aurora.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/aurora#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + " Arn
 = " + cluster.dbClusterArn()));
    }
}
```

- For API details, see DescribeDBClusters in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

CreateDBCluster

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

SDK for Java 2.x

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

```
    public static String createDBCluster(RdsClient rdsClient, String
 dbParameterGroupFamily, String dbName,
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();

            CreateDbClusterResponse response =
 rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see [CreateDBCluster](#) in *AWS SDK for Java 2.x API Reference*.

`CreateDBClusterParameterGroup`

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

SDK for Java 2.x

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

```
    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
 dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
 CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
 rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
 response.dbClusterParameterGroup().dbClusterParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
```

```
        }
    }
```

- For API details, see CreateDBClusterParameterGroup in *AWS SDK for Java 2.x API Reference*.

CreateDBClusterSnapshot

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

SDK for Java 2.x

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

```
    public static void createDBClusterSnapshot(RdsClient rdsClient, String
 dbInstanceClusterIdentifier,
            String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
 CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbClusterSnapshotResponse response =
 rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
 response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see CreateDBClusterSnapshot in *AWS SDK for Java 2.x API Reference*.

CreateDBInstance

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

SDK for Java 2.x

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

```
    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
 CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
 rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
 response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateDBInstance in *AWS SDK for Java 2.x API Reference*.

DeleteDBCluster

The following code example shows how to use DeleteDBCluster.

SDK for Java 2.x

Note

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

```
    public static void deleteCluster(RdsClient rdsClient, String
 dbInstanceClusterIdentifier) {
        try {
            DeleteDbClusterRequest deleteDbClusterRequest =
 DeleteDbClusterRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .skipFinalSnapshot(true)
                    .build();

            rdsClient.deleteDBCluster(deleteDbClusterRequest);
            System.out.println(dbInstanceClusterIdentifier + " was deleted!");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteDBCluster in *AWS SDK for Java 2.x API Reference*.

DeleteDBClusterParameterGroup

The following code example shows how to use DeleteDBClusterParameterGroup.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/rds#code-examples).

```
    public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
```

```
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();

            rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DeleteDBClusterParameterGroup] in *AWS SDK for Java 2.x API Reference*.

`DeleteDBInstance`

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

SDK for Java 2.x

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

```
    public static void deleteDatabaseInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
 DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.println("The status of the database is " +
 response.dbInstance().dbInstanceStatus());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteDBInstance in *AWS SDK for Java 2.x API Reference*.

DescribeDBClusterParameterGroups

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

SDK for Java 2.x

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

```
    public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
 dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
 DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
 rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
 group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
 group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeDBClusterParameterGroups in *AWS SDK for Java 2.x API Reference*.

`DescribeDBClusterParameters`

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

SDK for Java 2.x

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

```
    public static void describeDbClusterParameters(RdsClient rdsClient, String
 dbCLusterGroupName, int flag) {
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
 DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
 DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
```

```
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") == 0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeDBClusterParameters in *AWS SDK for Java 2.x API Reference*.

`DescribeDBClusterSnapshots`

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

SDK for Java 2.x

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

```
    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
```

```
            String dbInstanceClusterIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbClusterSnapshotsRequest snapshotsRequest =
 DescribeDbClusterSnapshotsRequest.builder()
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
 rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
 response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeDBClusterSnapshots in *AWS SDK for Java 2.x API Reference*.

DescribeDBClusters

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

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/rds#code-examples).

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
                       .build();
           }

           DescribeDbClusterParametersResponse response = rdsClient
                   .describeDBClusterParameters(dbParameterGroupsRequest);
           List<Parameter> dbParameters = response.parameters();
           String paraName;
           for (Parameter para : dbParameters) {
               // Only print out information about either auto_increment_offset or
               // auto_increment_increment.
               paraName = para.parameterName();
               if ((paraName.compareTo("auto_increment_offset") == 0)
                       || (paraName.compareTo("auto_increment_increment ") == 0)) {
                   System.out.println("*** The parameter name is  " + paraName);
                   System.out.println("*** The parameter value is  " +
para.parameterValue());
                   System.out.println("*** The parameter data type is " +
para.dataType());
                   System.out.println("*** The parameter description is " +
para.description());
```

```
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DescribeDBClusters](#) in *AWS SDK for Java 2.x API Reference*.

DescribeDBEngineVersions

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

SDK for Java 2.x

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

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
```

```
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeDBEngineVersions in *AWS SDK for Java 2.x API Reference*.

`DescribeDBInstances`

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

SDK for Java 2.x

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

```
    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
 dbClusterIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
 DescribeDbClustersRequest.builder()
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();
```

```
            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeDBInstances in *AWS SDK for Java 2.x API Reference*.

`DescribeOrderableDBInstanceOptions`

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

SDK for Java 2.x

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

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
```

```
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
engineOb.engine());
                System.out.println("The version number of the database engine " +
engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeOrderableDBInstanceOptions in *AWS SDK for Java 2.x API Reference*.

`ModifyDBClusterParameterGroup`

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

SDK for Java 2.x

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

```
    public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
 dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
 DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
 rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
 group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
 group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see ModifyDBClusterParameterGroup in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with DB clusters

The following code example shows how to:

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

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples).

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
```

```
 * 11. Creates a database instance in the cluster.
 * 12. Waits for DB instance to be ready.
 * 13. Creates a snapshot.
 * 14. Waits for DB snapshot to be ready.
 * 15. Deletes the DB cluster.
 * 16. Deletes the DB cluster group.
 */
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>"
                +
                "Where:\n" +
                "    dbClusterGroupName - The name of the DB cluster parameter
 group. \n" +
                "    dbParameterGroupFamily - The DB cluster parameter group family
 name (for example, aurora-mysql5.7). \n"
                +
                "    dbInstanceClusterIdentifier - The instance cluster identifier
 value.\n" +
                "    dbInstanceIdentifier - The database instance identifier.\n" +
                "    dbName - The database name.\n" +
                "    dbSnapshotIdentifier - The snapshot identifier.\n" +
                "    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials\"\n";
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];
```

```
        String secretName = args[6];

        // Retrieve the database credentials using AWS Secrets Manager.
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        String username = user.getUsername();
        String userPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Aurora example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
 dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBClusterParas(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Display the updated parameter value");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an Aurora DB cluster database");
        String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
 dbName, dbInstanceClusterIdentifier,
                username, userPassword);
        System.out.println("The ARN of the cluster is " + arnClusterVal);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of instance classes available for the
 selected engine");
        String instanceClass = getListInstanceClasses(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a database instance in the cluster.");
        String clusterDBARN = createDBInstanceCluster(rdsClient,
 dbInstanceIdentifier, dbInstanceClusterIdentifier,
                instanceClass);
        System.out.println("The ARN of the database is " + clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB instance to be ready");
        waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Create a snapshot");
```

```
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

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

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
```

```
                .build();

        GetSecretValueResponse valueResponse =
 secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void deleteDBClusterGroup(RdsClient rdsClient, String
 dbClusterGroupName, String clusterDBARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
 DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();
```

```
            rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteCluster(RdsClient rdsClient, String
 dbInstanceClusterIdentifier) {
        try {
            DeleteDbClusterRequest deleteDbClusterRequest =
 DeleteDbClusterRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .skipFinalSnapshot(true)
                    .build();

            rdsClient.deleteDBCluster(deleteDbClusterRequest);
            System.out.println(dbInstanceClusterIdentifier + " was deleted!");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
 DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.println("The status of the database is " +
 response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitForSnapshotReady(RdsClient rdsClient, String
 dbSnapshotIdentifier,
            String dbInstanceClusterIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbClusterSnapshotsRequest snapshotsRequest =
 DescribeDbClusterSnapshotsRequest.builder()
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
 rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
 response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
    public static void createDBClusterSnapshot(RdsClient rdsClient, String
 dbInstanceClusterIdentifier,
            String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
 CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbClusterSnapshotResponse response =
 rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
 response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitDBInstanceReady(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
 DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
```

```
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection 
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest = 
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response = 
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " + 
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String getListInstanceClasses(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest optionsRequest = 
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
```

```
                    .engine("aurora-mysql")
                    .maxRecords(20)
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(optionsRequest);
            List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
            String instanceClass = "";
            for (OrderableDBInstanceOption instanceOption : instanceOptions) {
                instanceClass = instanceOption.dbInstanceClass();
                System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
                System.out.println("The engine version is " +
instanceOption.engineVersion());
            }
            return instanceClass;

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
```

```
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBCluster(RdsClient rdsClient, String
 dbParameterGroupFamily, String dbName,
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();

            CreateDbClusterResponse response =
 rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
 dbParameterGroupFamily) {
        try {
```

```
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify the auto_increment_offset parameter.
    public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dClusterGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
            System.out.println(
```

```
                    "The parameter group " + response.dbClusterParameterGroupName()
 + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameters(RdsClient rdsClient, String
 dbCLusterGroupName, int flag) {
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
 DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
 DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") == 0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
 para.parameterValue());
                    System.out.println("*** The parameter data type is " +
 para.dataType());
                    System.out.println("*** The parameter description is " +
 para.description());
```

```
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
```

```
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
 rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
 response.dbClusterParameterGroup().dbClusterParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
 DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
 rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Auto Scaling examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Auto Scaling.

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

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

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

Get started

Hello Auto Scaling

The following code examples show how to get started using Auto Scaling.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/autoscale#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
 autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
```

```
        });
    }
}
```

- For API details, see DescribeAutoScalingGroups in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

CreateAutoScalingGroup

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
```

```
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName> <launchTemplateName> <serviceLinkedRoleARN>
 <vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
 instances in the Auto Scaling group can be created.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
 vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
            String groupName,
            String launchTemplateName,
            String vpcZoneId) {
```

```
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
 LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
 CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
                    .vpcZoneIdentifier(vpcZoneId)
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
                    .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

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

- For API details, see [CreateAutoScalingGroup] in *AWS SDK for Java 2.x API Reference*.

DeleteAutoScalingGroup

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/autoscale#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
 String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

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

- For API details, see DeleteAutoScalingGroup in *AWS SDK for Java 2.x API Reference*.

`DescribeAutoScalingGroups`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
```

```
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String instanceId = getAutoScaling(autoScalingClient, groupName);
        System.out.println(instanceId);
        autoScalingClient.close();
    }

    public static String getAutoScaling(AutoScalingClient autoScalingClient, String
 groupName) {
```

```
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
 group.autoScalingGroupName());
                System.out.println("The group ARN is " +
 group.autoScalingGroupARN());

                List<Instance> instances = group.instances();
                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                }
            }
            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see DescribeAutoScalingGroups in *AWS SDK for Java 2.x API Reference*.

DescribeAutoScalingInstances

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/autoscale#code-examples).

```
    public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
                    .builder()
                    .instanceIds(id)
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

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

- For API details, see DescribeAutoScalingInstances in *AWS SDK for Java 2.x API Reference*.

`DescribeScalingActivities`

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

SDK for Java 2.x

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

```
    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " + activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

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

- For API details, see DescribeScalingActivities in *AWS SDK for Java 2.x API Reference*.

DisableMetricsCollection

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

SDK for Java 2.x

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

```
    public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
 String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
 DisableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .build();


 autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
            System.out.println("The disable metrics collection operation was
 successful");

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

- For API details, see DisableMetricsCollection in *AWS SDK for Java 2.x API Reference*.

`EnableMetricsCollection`

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

SDK for Java 2.x

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

```
    public static void enableMetricsCollection(AutoScalingClient autoScalingClient, 
String groupName) {
        try {
            EnableMetricsCollectionRequest collectionRequest = 
EnableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .granularity("1Minute")
                    .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was 
successful");

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

- For API details, see EnableMetricsCollection in *AWS SDK for Java 2.x API Reference*.

`SetDesiredCapacity`

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

SDK for Java 2.x

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

```
    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                    .autoScalingGroupName(groupName)
                    .desiredCapacity(2)
                    .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");

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

- For API details, see [SetDesiredCapacity](#) in *AWS SDK for Java 2.x API Reference*.

`TerminateInstanceInAutoScalingGroup`

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

SDK for Java 2.x

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

```
    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
 autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
 TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);

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

- For API details, see [TerminateInstanceInAutoScalingGroup](#) in *AWS SDK for Java 2.x API Reference*.

`UpdateAutoScalingGroup`

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

SDK for Java 2.x

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

```
    public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
 String groupName,
            String launchTemplateName) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
 LaunchTemplateSpecification.builder()
```

```
                    .launchTemplateName(launchTemplateName)
                    .build();

            UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                    .maxSize(3)
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .build();

            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group  " +
groupName);

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

- For API details, see UpdateAutoScalingGroup in *AWS SDK for Java 2.x API Reference*.

Scenarios

Build and manage a resilient service

The following code example shows how to create a load-balanced web service that returns book, movie, and song recommendations. The example shows how the service responds to failures, and how to restructure the service for more resilience when failures occur.

- Use an Amazon EC2 Auto Scaling group to create Amazon Elastic Compute Cloud (Amazon EC2) instances based on a launch template and to keep the number of instances in a specified range.

- Handle and distribute HTTP requests with Elastic Load Balancing.
- Monitor the health of instances in an Auto Scaling group and forward requests only to healthy instances.
- Run a Python web server on each EC2 instance to handle HTTP requests. The web server responds with recommendations and health checks.
- Simulate a recommendation service with an Amazon DynamoDB table.
- Control web server response to requests and health checks by updating AWS Systems Manager parameters.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples).

Run the interactive scenario at a command prompt.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";
```

```
    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
 {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
 a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
```

```
                To keep things tidy and to avoid unwanted charges on your account,
 we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
 might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
 autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
 InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
```

```
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to create
 several AWS resources
                        to set up a load-balanced web service endpoint and explore
 some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
 provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances that
 each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
 across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
 targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
 tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
 instance starts.
                This script starts a Python web server defined in the `server.py`
 script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
 and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
 production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

                The template also defines an IAM policy that each instance uses to
 assume a role that grants
```

```
                permissions to access the DynamoDB recommendation table and Systems
Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
The target group
                defines how the load balancer connects to instances. The load
balancer provides a
```

```
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);
```

```
                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);
```

```
        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
 different parts of the system
                        to create situations where the web service fails, and shows
 how using a resilient
                        architecture can keep the web service running in spite of
 these failures.

                        At the start, the load balancer endpoint returns
 recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
 recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
 named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
 set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer endpoint
 returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
 don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
 the web service can return a static response.
                       While this is not a perfect solution, it presents the
 customer with a somewhat better experience than failure.
                       """);
```

```
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns a
 static response.
                The service still reports as healthy because health checks are still
 shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
 the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance id
 value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
 badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
 badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
 profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
 returns either a recommendation or a static response,
                        depending on which instance is selected by the load
 balancer.
                        """);
```

```
        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);
```

```
        System.out.println("""
                Even while the instance is terminating and the new instance is 
starting, sending a GET
                request to the web service continues to get a successful 
recommendation response because
                the load balancer routes requests to the healthy instances. After 
the replacement instance
                starts and reports as healthy, it is included in the load balancing 
rotation.
                Note that terminating and replacing an instance typically takes 
several minutes, during which time you
                can see the changing health check status until the new instance is 
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean 
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException, 
InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting 
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
```

```
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
 loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
 HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
 loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
 InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
 targets:\n");
```

```
                        List<TargetHealthDescription> health =
 loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
 check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Create a class that wraps Auto Scaling and Amazon EC2 actions.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
```

```
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
```

```
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                                // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
```

```
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with profile
 %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
 waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
 getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
 informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
```

```
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
```

```
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
 rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
 attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
 name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
```

```
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();
```

```
            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
```

```
        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

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

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
```

```
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
```

```
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
 DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
 getAutoScalingClient().describeAutoScalingGroups(request);
```

```
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
 DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
```

```
listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
```

```
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Create a class that wraps Elastic Load Balancing actions.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
```

```
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
```

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

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
 describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
 builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
 InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
```

```
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
```

```
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
```

```
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .defaultActions(action)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Create a class that uses DynamoDB to simulate a recommendation service.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
```

```
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
```

```
                                .attributeName("ItemId")
                                .attributeType(ScalarAttributeType.N)
                                .build())
                .keySchema(
                        KeySchemaElement.builder()
                                .attributeName("MediaType")
                                .keyType(KeyType.HASH)
                                .build(),
                        KeySchemaElement.builder()
                                .attributeName("ItemId")
                                .keyType(KeyType.RANGE)
                                .build())
                .provisionedThroughput(
                        ProvisionedThroughput.builder()
                                .readCapacityUnits(5L)
                                .writeCapacityUnits(5L)
                                .build())
                .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
```

```
    public void populateTable(String fileName, String tableName) throws IOException
 {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Create a class that wraps Systems Manager actions.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
```

```
            put(failureResponse, "none");
            put(healthCheck, "shallow");
        }

        public void put(String name, String value) {
            SsmClient ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();

            PutParameterRequest parameterRequest = PutParameterRequest.builder()
                    .name(name)
                    .value(value)
                    .overwrite(true)
                    .type("String")
                    .build();

            ssmClient.putParameter(parameterRequest);
            System.out.printf("Setting demo parameter %s to '%s'.", name, value);
        }
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones

- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Manage groups and instances

The following code example shows how to:

- Create an Amazon EC2 Auto Scaling group with a launch template and Availability Zones, and get information about running instances.
- Enable Amazon CloudWatch metrics collection.
- Update the group's desired capacity and wait for an instance to start.
- Terminate an instance in the group.
- List scaling activities that occur in response to user requests and capacity changes.
- Get statistics for CloudWatch metrics, then clean up resources.

SDK for Java 2.x

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

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <groupName> <launchTemplateName> <vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
 instances in the Auto Scaling group can be created.
                """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example
 scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an Auto Scaling group named " + groupName);
        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
 vpcZoneId);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get Auto Scale group Id value");
        String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
 groupName);
        if (instanceId.compareTo("") == 0) {
            System.out.println("Error - no instance Id value");
            System.exit(1);
        } else {
            System.out.println("The instance Id value is " + instanceId);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe Auto Scaling with the Id value " +
 instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to
 3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for
 the group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
```

```
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient
 autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
 DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " + activity.activityId());
                System.out.println("The activity details are " +
 activity.details());
            }

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

```
    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                    .autoScalingGroupName(groupName)
                    .desiredCapacity(2)
                    .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");

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

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
            String groupName,
            String launchTemplateName,
            String vpcZoneId) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
                    .vpcZoneIdentifier(vpcZoneId)
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();
```

```
            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

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

    public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
                    .builder()
                    .instanceIds(id)
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

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

    public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
```

```
                    .maxRecords(10)
                    .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("*** The service to use for the health checks: "
+ group.healthCheckType());
            }

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

    public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                    System.out.println("The lifecycle state is " +
instance.lifecycleState());
                }
            }
```

```
            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .granularity("1Minute")
                    .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was
successful");

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

    public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
            System.out.println("The disable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }

    public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
        try {
            DescribeAccountLimitsResponse response =
 autoScalingClient.describeAccountLimits();
            System.out.println("The max number of auto scaling groups is " +
 response.maxNumberOfAutoScalingGroups());
            System.out.println("The current number of auto scaling groups is " +
 response.numberOfAutoScalingGroups());

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

    public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
 String groupName,
            String launchTemplateName) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
 LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            UpdateAutoScalingGroupRequest groupRequest =
 UpdateAutoScalingGroupRequest.builder()
                    .maxSize(3)
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .build();

            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
```

```
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group  " +
groupName);

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

    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);

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

    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

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

```
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Amazon Bedrock examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Bedrock.

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

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

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

Actions

GetFoundationModel

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

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/bedrock#code-examples).

Get details about a foundation model using the synchronous Amazon Bedrock client.

```
    /**
     * Get details about an Amazon Bedrock foundation model.
     *
     * @param bedrockClient   The service client for accessing Amazon Bedrock.
     * @param modelIdentifier The model identifier.
     * @return An object containing the foundation model's details.
     */
    public static FoundationModelDetails getFoundationModel(BedrockClient
 bedrockClient, String modelIdentifier) {
        try {
            GetFoundationModelResponse response = bedrockClient.getFoundationModel(
                    r -> r.modelIdentifier(modelIdentifier)
            );

            FoundationModelDetails model = response.modelDetails();

            System.out.println(" Model ID:                     " + model.modelId());
            System.out.println(" Model ARN:                    " +
 model.modelArn());
            System.out.println(" Model Name:                   " +
 model.modelName());
            System.out.println(" Provider Name:                " +
 model.providerName());
            System.out.println(" Lifecycle status:             " +
 model.modelLifecycle().statusAsString());
            System.out.println(" Input modalities:             " +
 model.inputModalities());
```

```
            System.out.println(" Output modalities:            " +
model.outputModalities());
            System.out.println(" Supported customizations:     " +
model.customizationsSupported());
            System.out.println(" Supported inference types:    " +
model.inferenceTypesSupported());
            System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

            return model;

        } catch (ValidationException e) {
            throw new IllegalArgumentException(e.getMessage());
        } catch (SdkException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

Get details about a foundation model using the asynchronous Amazon Bedrock client.

```
    /**
     * Get details about an Amazon Bedrock foundation model.
     *
     * @param bedrockClient   The async service client for accessing Amazon Bedrock.
     * @param modelIdentifier The model identifier.
     * @return An object containing the foundation model's details.
     */
    public static FoundationModelDetails getFoundationModel(BedrockAsyncClient
bedrockClient, String modelIdentifier) {
        try {
            CompletableFuture<GetFoundationModelResponse> future =
bedrockClient.getFoundationModel(
                    r -> r.modelIdentifier(modelIdentifier)
            );

            FoundationModelDetails model = future.get().modelDetails();

            System.out.println(" Model ID:                     " + model.modelId());
            System.out.println(" Model ARN:                    " +
model.modelArn());
```

```
            System.out.println(" Model Name:                 " +
model.modelName());
            System.out.println(" Provider Name:              " +
model.providerName());
            System.out.println(" Lifecycle status:           " +
model.modelLifecycle().statusAsString());
            System.out.println(" Input modalities:           " +
model.inputModalities());
            System.out.println(" Output modalities:          " +
model.outputModalities());
            System.out.println(" Supported customizations:     " +
model.customizationsSupported());
            System.out.println(" Supported inference types:    " +
model.inferenceTypesSupported());
            System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

            return model;

        } catch (ExecutionException e) {
            if (e.getMessage().contains("ValidationException")) {
                throw new IllegalArgumentException(e.getMessage());
            } else {
                System.err.println(e.getMessage());
                throw new RuntimeException(e);
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

- For API details, see GetFoundationModel in *AWS SDK for Java 2.x API Reference*.

`ListFoundationModels`

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

SDK for Java 2.x

Note

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

List the available Amazon Bedrock foundation models using the synchronous Amazon Bedrock client.

```
    /**
     * Lists Amazon Bedrock foundation models that you can use.
     * You can filter the results with the request parameters.
     *
     * @param bedrockClient The service client for accessing Amazon Bedrock.
     * @return A list of objects containing the foundation models' details
     */
    public static List<FoundationModelSummary> listFoundationModels(BedrockClient
 bedrockClient) {

        try {
            ListFoundationModelsResponse response =
 bedrockClient.listFoundationModels(r -> {});

            List<FoundationModelSummary> models = response.modelSummaries();

            if (models.isEmpty()) {
                System.out.println("No available foundation models in " +
 region.toString());
            } else {
                for (FoundationModelSummary model : models) {
                    System.out.println("Model ID: " + model.modelId());
                    System.out.println("Provider: " + model.providerName());
                    System.out.println("Name:     " + model.modelName());
                    System.out.println();
                }
            }

            return models;

        } catch (SdkClientException e) {
```

```
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

List the available Amazon Bedrock foundation models using the asynchronous Amazon Bedrock client.

```
    /**
     * Lists Amazon Bedrock foundation models that you can use.
     * You can filter the results with the request parameters.
     *
     * @param bedrockClient The async service client for accessing Amazon Bedrock.
     * @return A list of objects containing the foundation models' details
     */
    public static List<FoundationModelSummary>
 listFoundationModels(BedrockAsyncClient bedrockClient) {
        try {
            CompletableFuture<ListFoundationModelsResponse> future =
 bedrockClient.listFoundationModels(r -> {});

            List<FoundationModelSummary> models = future.get().modelSummaries();

            if (models.isEmpty()) {
                System.out.println("No available foundation models in " +
 region.toString());
            } else {
                for (FoundationModelSummary model : models) {
                    System.out.println("Model ID: " + model.modelId());
                    System.out.println("Provider: " + model.providerName());
                    System.out.println("Name:     " + model.modelName());
                    System.out.println();
                }
            }

            return models;

        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        } catch (ExecutionException e) {
```

```
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

- For API details, see ListFoundationModels in *AWS SDK for Java 2.x API Reference*.

Amazon Bedrock Runtime examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Bedrock Runtime.

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

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

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

Topics

AI21 Labs Jurassic-2

Converse

The following code example shows how to send a text message to AI21 Labs Jurassic-2, using Bedrock's Converse API.

SDK for Java 2.x

Note

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

Send a text message to AI21 Labs Jurassic-2, using Bedrock's Converse API.

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";
```

```
        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converse();
    }
}
```

Send a text message to AI21 Labs Jurassic-2, using Bedrock's Converse API with the async Java client.

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2
```

```
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
```

```
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
 response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- For API details, see Converse in *AWS SDK for Java 2.x API Reference*.

InvokeModel

The following code example shows how to send a text message to AI21 Labs Jurassic-2, using the Invoke Model API.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/bedrock-runtime#code-examples).

Use the Invoke Model API to send a text message.

```
// Use the native inference API to send a text message to AI21 Labs Jurassic-2.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
jurassic2.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

        // Define the prompt for the model.
```

```
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/completions/0/data/
text").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

Amazon Titan Image Generator

InvokeModel

The following code example shows how to invoke Amazon Titan Image on Amazon Bedrock to generate an image.

SDK for Java 2.x

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

Create an image with the Amazon Titan Image Generator.

```
// Create an image with the Amazon Titan Image Generator.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Image G1.
        var modelId = "amazon.titan-image-generator-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
```

```
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html
        var nativeRequestTemplate = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": { "text": "{{prompt}}" },
                    "imageGenerationConfig": { "seed": {{seed}} }
                }""";

        // Define the prompt for the image generation.
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random 31-bit seed for the image generation (max. 2,147,483,647).
        var seed = new BigInteger(31, new SecureRandom());

        // Embed the prompt and seed in the model's native request payload.
        var nativeRequest = nativeRequestTemplate
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", seed.toString());

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated image data from the model's response.
            var base64ImageData = new JSONPointer("/
images/0").queryFrom(responseBody).toString();

            return base64ImageData;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
```

```
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }
}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

Amazon Titan Text

Converse

The following code example shows how to send a text message to Amazon Titan Text, using Bedrock's Converse API.

SDK for Java 2.x

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

Send a text message to Amazon Titan Text, using Bedrock's Converse API.

```
// Use the Converse API to send a text message to Amazon Titan Text.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        converse();
    }
}
```

Send a text message to Amazon Titan Text, using Bedrock's Converse API with the async Java client.

```
// Use the Converse API to send a text message to Amazon Titan Text
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
```

```
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
 response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        converseAsync();
    }
}
```

- For API details, see [Converse](#) in *AWS SDK for Java 2.x API Reference*.

ConverseStream

The following code example shows how to send a text message to Amazon Titan Text, using Bedrock's Converse API and process the response stream in real-time.

SDK for Java 2.x

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

Send a text message to Amazon Titan Text, using Bedrock's Converse API and process the response stream in real-time.

```
// Use the Converse API to send a text message to Amazon Titan Text
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
handler.
            client.converseStream(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();
```

```
        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- For API details, see [ConverseStream](#) in *AWS SDK for Java 2.x API Reference*.

InvokeModel

The following code example shows how to send a text message to Amazon Titan Text, using the Invoke Model API.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message.

```
// Use the native inference API to send a text message to Amazon Titan Text.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
```

```
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/results/0/
outputText").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        invokeModel();
    }
}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

InvokeModelWithResponseStream

The following code example shows how to send a text message to Amazon Titan Text models, using the Invoke Model API, and print the response stream.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message and process the response stream in real-time.

```
// Use the native inference API to send a text message to Amazon Titan Text
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH
```

```
public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
 InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        // Create a request with the model ID and the model's native request
 payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
                .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
 real-time.
        var responseStreamHandler =
 InvokeModelWithResponseStreamResponseHandler.builder()
```

```
                .subscriber(Visitor.builder().onChunk(chunk -> {
                    // Extract and print the text from the model's native response.
                    var response = new JSONObject(chunk.bytes().asUtf8String());
                    var text = new JSONPointer("/outputText").queryFrom(response);
                    System.out.print(text);

                    // Append the text to the response text buffer.
                    completeResponseTextBuffer.append(text);
                }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
 responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) throws ExecutionException,
 InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- For API details, see [InvokeModelWithResponseStream] in *AWS SDK for Java 2.x API Reference*.

Amazon Titan Text Embeddings

InvokeModel

The following code example shows how to:

- Get started creating your first embedding.
- Create embeddings configuring the number of dimensions and normalization (V2 only).

SDK for Java 2.x

Note

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

Create your first embedding with Titan Text Embeddings V2.

```
// Generate and print an embedding with Amazon Titan Text Embeddings.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Text Embeddings V2.
        var modelId = "amazon.titan-embed-text-v2:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-embed-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{inputText}}\" }";

        // The text to convert into an embedding.
```

```
        var inputText = "Please recommend books with a theme similar to the movie
 'Inception'.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{inputText}}",
 inputText);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/
embedding").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

Invoke Titan Text Embeddings V2 configuring the number of dimensions and normalization.

```
    /**
     * Invoke Amazon Titan Text Embeddings V2 with additional inference parameters.
     *
```

```
     * @param inputText  - The text to convert to an embedding.
     * @param dimensions - The number of dimensions the output embeddings should
 have.
     *                    Values accepted by the model: 256, 512, 1024.
     * @param normalize  - A flag indicating whether or not to normalize the output
 embeddings.
     * @return The {@link JSONObject} representing the model's response.
     */
    public static JSONObject invokeModel(String inputText, int dimensions, boolean
 normalize) {

        // Create a Bedrock Runtime client in the AWS Region of your choice.
        var client = BedrockRuntimeClient.builder()
                .region(Region.US_WEST_2)
                .build();

        // Set the model ID, e.g., Titan Embed Text v2.0.
        var modelId = "amazon.titan-embed-text-v2:0";

        // Create the request for the model.
        var nativeRequest = """
                {
                    "inputText": "%s",
                    "dimensions": %d,
                    "normalize": %b
                }
                """.formatted(inputText, dimensions, normalize);

        // Encode and send the request.
        var response = client.invokeModel(request -> {
            request.body(SdkBytes.fromUtf8String(nativeRequest));
            request.modelId(modelId);
        });

        // Decode the model's response.
        var modelResponse = new JSONObject(response.body().asUtf8String());

        // Extract and print the generated embedding and the input text token count.
        var embedding = modelResponse.getJSONArray("embedding");
        var inputTokenCount = modelResponse.getBigInteger("inputTextTokenCount");
        System.out.println("Embedding: " + embedding);
        System.out.println("\nInput token count: " + inputTokenCount);

        // Return the model's native response.
```

```
        return modelResponse;
    }
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

Anthropic Claude

Converse

The following code example shows how to send a text message to Anthropic Claude, using Bedrock's Converse API.

SDK for Java 2.x

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

Send a text message to Anthropic Claude, using Bedrock's Converse API.

```
// Use the Converse API to send a text message to Anthropic Claude.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
```

```
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converse();
    }
}
```

Send a text message to Anthropic Claude, using Bedrock's Converse API with the async Java client.

```
// Use the Converse API to send a text message to Anthropic Claude
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
```

```
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
 response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- For API details, see Converse in *AWS SDK for Java 2.x API Reference*.

ConverseStream

The following code example shows how to send a text message to Anthropic Claude, using Bedrock's Converse API and process the response stream in real-time.

SDK for Java 2.x

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

Send a text message to Anthropic Claude, using Bedrock's Converse API and process the response stream in real-time.

```
// Use the Converse API to send a text message to Anthropic Claude
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
```

```
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
 handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- For API details, see ConverseStream in *AWS SDK for Java 2.x API Reference*.

InvokeModel

The following code example shows how to send a text message to Anthropic Claude, using the Invoke Model API.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message.

```
// Use the native inference API to send a text message to Anthropic Claude.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
```

```
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
anthropic-claude-messages.html
        var nativeRequestTemplate = """
                {
                    "anthropic_version": "bedrock-2023-05-31",
                    "max_tokens": 512,
                    "temperature": 0.5,
                    "messages": [{
                        "role": "user",
                        "content": "{{prompt}}"
                    }]
                }""";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/content/0/
text").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
```

```
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

InvokeModelWithResponseStream

The following code example shows how to send a text message to Anthropic Claude models, using the Invoke Model API, and print the response stream.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message and process the response stream in real-time.

```
// Use the native inference API to send a text message to Anthropic Claude
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

import java.util.Objects;
```

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

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
 InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
anthropic-claude-messages.html
        var nativeRequestTemplate = """
                {
                    "anthropic_version": "bedrock-2023-05-31",
                    "max_tokens": 512,
                    "temperature": 0.5,
                    "messages": [{
                        "role": "user",
                        "content": "{{prompt}}"
                    }]
                }""";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);
```

```
        // Create a request with the model ID and the model's native request
 payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
                .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
 real-time.
        var responseStreamHandler =
 InvokeModelWithResponseStreamResponseHandler.builder()
                .subscriber(Visitor.builder().onChunk(chunk -> {
                    var response = new JSONObject(chunk.bytes().asUtf8String());

                    // Extract and print the text from the content blocks.
                    if (Objects.equals(response.getString("type"),
 "content_block_delta")) {
                        var text = new JSONPointer("/delta/
text").queryFrom(response);
                        System.out.print(text);

                        // Append the text to the response text buffer.
                        completeResponseTextBuffer.append(text);
                    }
                }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
 responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) throws ExecutionException,
 InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- For API details, see InvokeModelWithResponseStream in *AWS SDK for Java 2.x API Reference*.

Cohere Command

Converse

The following code example shows how to send a text message to Cohere Command, using Bedrock's Converse API.

SDK for Java 2.x

Note

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

Send a text message to Cohere Command, using Bedrock's Converse API.

```
// Use the Converse API to send a text message to Cohere Command.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
```

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

Send a text message to Cohere Command, using Bedrock's Converse API with the async Java client.

```
// Use the Converse API to send a text message to Cohere Command
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
```

```
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
 response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
```

```
}
```

- For API details, see Converse in *AWS SDK for Java 2.x API Reference*.

ConverseStream

The following code example shows how to send a text message to Cohere Command, using Bedrock's Converse API and process the response stream in real-time.

SDK for Java 2.x

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

Send a text message to Cohere Command, using Bedrock's Converse API and process the response stream in real-time.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
```

```
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
 handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
```

```
    }
}
```

- For API details, see ConverseStream in *AWS SDK for Java 2.x API Reference*.

InvokeModel: Command R and R+

The following code example shows how to send a text message to Cohere Command R and R+, using the Invoke Model API.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message.

```
// Use the native inference API to send a text message to Cohere Command R.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Command_R_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();
```

```
        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command-r-plus.html
        var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/text").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

InvokeModel: Command and Command Light

The following code example shows how to send a text message to Cohere Command, using the Invoke Model API.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message.

```
// Use the native inference API to send a text message to Cohere Command.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Command_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command Light.
```

```
        var modelId = "cohere.command-light-text-v14";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/generations/0/
text").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

InvokeModelWithResponseStream: Command R and R+

The following code example shows how to send a text message to Cohere Command, using the Invoke Model API with a response stream.

SDK for Java 2.x

Note

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

Use the Invoke Model API to send a text message and process the response stream in real-time.

```
// Use the native inference API to send a text message to Cohere Command R
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class Command_R_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
 InterruptedException {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command-r-plus.html
        var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        // Create a request with the model ID and the model's native request
 payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
                .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
 real-time.
        var responseStreamHandler =
 InvokeModelWithResponseStreamResponseHandler.builder()
                .subscriber(Visitor.builder().onChunk(chunk -> {
                    // Extract and print the text from the model's native response.
                    var response = new JSONObject(chunk.bytes().asUtf8String());
                    var text = new JSONPointer("/text").queryFrom(response);
                    System.out.print(text);
```

```
                    // Append the text to the response text buffer.
                    completeResponseTextBuffer.append(text);
                }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
 responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) throws ExecutionException,
 InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

InvokeModelWithResponseStream: Command and Command Light

The following code example shows how to send a text message to Cohere Command, using the Invoke Model API with a response stream.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message and process the response stream in real-time.

```
// Use the native inference API to send a text message to Cohere Command
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class Command_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
 InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command Light.
        var modelId = "cohere.command-light-text-v14";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";
```

```
        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        // Create a request with the model ID and the model's native request
 payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
                .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
 real-time.
        var responseStreamHandler =
 InvokeModelWithResponseStreamResponseHandler.builder()
                .subscriber(Visitor.builder().onChunk(chunk -> {
                    // Extract and print the text from the model's native response.
                    var response = new JSONObject(chunk.bytes().asUtf8String());
                    var text = new JSONPointer("/generations/0/
text").queryFrom(response);
                    System.out.print(text);

                    // Append the text to the response text buffer.
                    completeResponseTextBuffer.append(text);
                }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
 responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        }
```

```
    }

    public static void main(String[] args) throws ExecutionException,
 InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

Meta Llama

Converse

The following code example shows how to send a text message to Meta Llama, using Bedrock's Converse API.

SDK for Java 2.x

Note

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

Send a text message to Meta Llama, using Bedrock's Converse API.

```
// Use the Converse API to send a text message to Meta Llama.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        converse();
    }
}
```

Send a text message to Meta Llama, using Bedrock's Converse API with the async Java client.

```
// Use the Converse API to send a text message to Meta Llama
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
```

```
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText = 
response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated 
text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
```

```
    }
}
```

- For API details, see Converse in *AWS SDK for Java 2.x API Reference*.

ConverseStream

The following code example shows how to send a text message to Meta Llama, using Bedrock's Converse API and process the response stream in real-time.

SDK for Java 2.x

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

Send a text message to Meta Llama, using Bedrock's Converse API and process the response stream in real-time.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
 handler.
            client.converseStream(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
```

```
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- For API details, see [ConverseStream](#) in *AWS SDK for Java 2.x API Reference*.

InvokeModel: Llama 2

The following code example shows how to send a text message to Meta Llama 2, using the Invoke Model API.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message.

```
// Use the native inference API to send a text message to Meta Llama 2.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama2_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
```

```
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 2 Chat 13B.
        var modelId = "meta.llama2-13b-chat-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 2's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
 prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
 instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
```

```
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

InvokeModel: Llama 3

The following code example shows how to send a text message to Meta Llama 3, using the Invoke Model API.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message.

```
// Use the native inference API to send a text message to Meta Llama 3.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama3_InvokeModel {

    public static String invokeModel() {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 3's instruction format.
        var instruction = (
                "<|begin_of_text|>\\n" +
                "<|start_header_id|>user<|end_header_id|>\\n" +
                "{{prompt}} <|eot_id|>\\n" +
                "<|start_header_id|>assistant<|end_header_id|>\\n"
        ).replace("{{prompt}}", prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
 instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());
```

```
            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

InvokeModelWithResponseStream: Llama 2

The following code example shows how to send a text message to Meta Llama 2, using the Invoke Model API, and print the response stream.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message and process the response stream in real-time.

```
// Use the native inference API to send a text message to Meta Llama 2
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class Llama2_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
 InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 2 Chat 13B.
        var modelId = "meta.llama2-13b-chat-v1";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 2's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
 prompt);
```

```
        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
 instruction);

        // Create a request with the model ID and the model's native request
 payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
                .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
 real-time.
        var responseStreamHandler =
 InvokeModelWithResponseStreamResponseHandler.builder()
                .subscriber(Visitor.builder().onChunk(chunk -> {
                    // Extract and print the text from the model's native response.
                    var response = new JSONObject(chunk.bytes().asUtf8String());
                    var text = new JSONPointer("/generation").queryFrom(response);
                    System.out.print(text);

                    // Append the text to the response text buffer.
                    completeResponseTextBuffer.append(text);
                }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
 responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) throws ExecutionException,
 InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- For API details, see [InvokeModelWithResponseStream] in *AWS SDK for Java 2.x API Reference*.

InvokeModelWithResponseStream: Llama 3

The following code example shows how to send a text message to Meta Llama 3, using the Invoke Model API, and print the response stream.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message and process the response stream in real-time.

```
// Use the native inference API to send a text message to Meta Llama 3
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH
```

```
public class Llama3_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
 InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 3's instruction format.
        var instruction = (
                "<|begin_of_text|>\\n" +
                "<|start_header_id|>user<|end_header_id|>\\n" +
                "{{prompt}} <|eot_id|>\\n" +
                "<|start_header_id|>assistant<|end_header_id|>\\n"
        ).replace("{{prompt}}", prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
 instruction);

        // Create a request with the model ID and the model's native request
 payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
```

```
                .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
 real-time.
        var responseStreamHandler =
 InvokeModelWithResponseStreamResponseHandler.builder()
                .subscriber(Visitor.builder().onChunk(chunk -> {
                    // Extract and print the text from the model's native response.
                    var response = new JSONObject(chunk.bytes().asUtf8String());
                    var text = new JSONPointer("/generation").queryFrom(response);
                    System.out.print(text);

                    // Append the text to the response text buffer.
                    completeResponseTextBuffer.append(text);
                }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
 responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) throws ExecutionException,
 InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- For API details, see InvokeModelWithResponseStream in *AWS SDK for Java 2.x API Reference*.

Mistral AI

Converse

The following code example shows how to send a text message to Mistral, using Bedrock's Converse API.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/bedrock-runtime#code-examples).

Send a text message to Mistral, using Bedrock's Converse API.

```
// Use the Converse API to send a text message to Mistral.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";
```

```
        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }

    }

    public static void main(String[] args) {
        converse();
    }
}
```

Send a text message to Mistral, using Bedrock's Converse API with the async Java client.

```
// Use the Converse API to send a text message to Mistral
```

```
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

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

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
```

```
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
 response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- For API details, see [Converse](#) in *AWS SDK for Java 2.x API Reference*.

ConverseStream

The following code example shows how to send a text message to Mistral, using Bedrock's Converse API and process the response stream in real-time.

SDK for Java 2.x

Note

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

Send a text message to Mistral, using Bedrock's Converse API and process the response stream in real-time.

```
// Use the Converse API to send a text message to Mistral
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
```

```
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
 handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- For API details, see ConverseStream in *AWS SDK for Java 2.x API Reference*.

InvokeModel

The following code example shows how to send a text message to Mistral models, using the Invoke Model API.

SDK for Java 2.x

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

Use the Invoke Model API to send a text message.

```
// Use the native inference API to send a text message to Mistral.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
```

```
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
mistral-text-completion.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Mistral's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
 prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
 instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/outputs/0/
text").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

InvokeModelWithResponseStream

The following code example shows how to send a text message to Mistral AI models, using the Invoke Model API, and print the response stream.

SDK for Java 2.x

Note

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

Use the Invoke Model API to send a text message and process the response stream in real-time.

```
// Use the native inference API to send a text message to Mistral
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
 InterruptedException {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
mistral-text-completion.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Mistral's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
 prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
 instruction);

        // Create a request with the model ID and the model's native request
 payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
                .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
 real-time.
        var responseStreamHandler =
 InvokeModelWithResponseStreamResponseHandler.builder()
```

```
                .subscriber(Visitor.builder().onChunk(chunk -> {
                    // Extract and print the text from the model's native response.
                    var response = new JSONObject(chunk.bytes().asUtf8String());
                    var text = new JSONPointer("/outputs/0/
text").queryFrom(response);
                    System.out.print(text);

                    // Append the text to the response text buffer.
                    completeResponseTextBuffer.append(text);
                }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
 responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) throws ExecutionException,
 InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- For API details, see InvokeModelWithResponseStream in *AWS SDK for Java 2.x API Reference*.

Scenarios

Create a playground application to interact with Amazon Bedrock foundation models

The following code example shows how to create playgrounds to interact with Amazon Bedrock foundation models through different modalities.

SDK for Java 2.x

The Java Foundation Model (FM) Playground is a Spring Boot sample application that showcases how to use Amazon Bedrock with Java. This example shows how Java developers can use Amazon Bedrock to build generative AI-enabled applications. You can test and interact with Amazon Bedrock foundation models by using the following three playgrounds:

- A text playground.
- A chat playground.
- An image playground.

The example also lists and displays the foundation models you have access to, along with their characteristics. For source code and deployment instructions, see the project in GitHub.

Services used in this example

- Amazon Bedrock Runtime

Stable Diffusion

InvokeModel

The following code example shows how to invoke Stability.ai Stable Diffusion XL on Amazon Bedrock to generate an image.

SDK for Java 2.x

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

Create an image with Stable Diffusion.

```
// Create an image with Stable Diffusion.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Stable Diffusion XL v1.
        var modelId = "stability.stable-diffusion-xl-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
diffusion-1-0-text-image.html
        var nativeRequestTemplate = """
                {
                    "text_prompts": [{ "text": "{{prompt}}" }],
                    "style_preset": "{{style}}",
                    "seed": {{seed}}
                }""";

        // Define the prompt for the image generation.
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random 32-bit seed for the image generation (max. 4,294,967,295).
        var seed = new BigInteger(31, new SecureRandom());

        // Choose a style preset.
        var style = "cinematic";
```

```
        // Embed the prompt, seed, and style in the model's native request payload.
        String nativeRequest = nativeRequestTemplate
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", seed.toString())
                .replace("{{style}}", style);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated image data from the model's response.
            var base64ImageData = new JSONPointer("/artifacts/0/base64")
                    .queryFrom(responseBody)
                    .toString();

            return base64ImageData;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }


}
```

- For API details, see InvokeModel in *AWS SDK for Java 2.x API Reference*.

CloudFront examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudFront.

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

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

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

`CreateDistribution`

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

SDK for Java 2.x

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

The following example uses an Amazon Simple Storage Service (Amazon S3) bucket as a content origin.

After creating the distribution, the code creates a CloudFrontWaiter to wait until the distribution is deployed before returning the distribution.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

        private static final Logger logger =
 LoggerFactory.getLogger(CreateDistribution.class);

        public static Distribution createDistribution(CloudFrontClient
 cloudFrontClient, S3Client s3Client,
                        final String bucketName, final String keyGroupId, final
 String originAccessControlId) {

                final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                                .get("x-amz-bucket-region").get(0);
                final String originDomain = bucketName + ".s3." + region +
 ".amazonaws.com";
                String originId = originDomain; // Use the originDomain value for
 the originId.

                // The service API requires some deprecated methods, such as
                // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
                CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                                .distributionConfig(b1 -> b1
                                                .origins(b2 -> b2
                                                                .quantity(1)
                                                                .items(b3 -> b3

 .domainName(originDomain)
```

```
 .id(originId)

 .s3OriginConfig(builder4 -> builder4

             .originAccessIdentity(

                                 ""))

 .originAccessControlId(

             originAccessControlId)))
                                                .defaultCacheBehavior(b2 -> b2

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

 .targetOriginId(originId)
                                                                .minTTL(200L)
                                                                .forwardedValues(b5
 -> b5

 .cookies(cp -> cp

             .forward(ItemSelection.NONE))

 .queryString(true))
                                                                .trustedKeyGroups(b3
 -> b3

 .quantity(1)

 .items(keyGroupId)

 .enabled(true))
                                                                .allowedMethods(b4 -
> b4

 .quantity(2)

 .items(Method.HEAD, Method.GET)

 .cachedMethods(b5 -> b5

             .quantity(2)
```

```
            .items(Method.HEAD,

                             Method.GET))))
                                                 .cacheBehaviors(b -> b
                                                                  .quantity(1)
                                                                  .items(b2 -> b2

.pathPattern("/index.html")

.viewerProtocolPolicy(

            ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3

            .quantity(1)

            .items(keyGroupId)

            .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4

            .cookies(cp -> cp

                             .forward(ItemSelection.NONE))

            .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)

            .items(Method.HEAD,

                             Method.GET)

            .cachedMethods(b6 -> b6

                             .quantity(2)
```

```
                                .items(Method.HEAD,

                                                Method.GET)))))
                                                    .enabled(true)
                                                    .comment("Distribution built with
 java")

 .callerReference(Instant.now().toString())));

                final Distribution distribution = createDistResponse.distribution();
                logger.info("Distribution created. DomainName: [{}]  Id: [{}]",
 distribution.domainName(),
                                distribution.id());
                logger.info("Waiting for distribution to be deployed ...");
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                                        .matched();
                        responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Distribution not created"));
                        logger.info("Distribution deployed. DomainName: [{}]  Id:
 [{}]", distribution.domainName(),
                                        distribution.id());
                }
                return distribution;
        }
}
```

- For API details, see [CreateDistribution](#) in *AWS SDK for Java 2.x API Reference*.

`CreateFunction`

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

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <functionName> <filePath>

                Where:
                    functionName - The name of the function to create.\s
                    filePath - The path to a file that contains the application
 logic for the function.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        String funArn = createNewFunction(cloudFrontClient, functionName, filePath);
        System.out.println("The function ARN is " + funArn);
        cloudFrontClient.close();
    }

    public static String createNewFunction(CloudFrontClient cloudFrontClient, String
 functionName, String filePath) {
        try {
            InputStream fileIs =
 CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

            FunctionConfig config = FunctionConfig.builder()
                    .comment("Created by using the CloudFront Java API")
                    .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
                    .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                    .name(functionName)
                    .functionCode(functionCode)
                    .functionConfig(config)
                    .build();

            CreateFunctionResponse response =
 cloudFrontClient.createFunction(functionRequest);
            return response.functionSummary().functionMetadata().functionARN();

        } catch (CloudFrontException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see [CreateFunction] in *AWS SDK for Java 2.x API Reference*.

CreateKeyGroup

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

SDK for Java 2.x

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

A key group requires at least one public key that is used to verify signed URLs or cookies.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger =
 LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
 publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b -> b.keyGroupConfig(c
 -> c
                .items(publicKeyId)
                .name("JavaKeyGroup" + UUID.randomUUID())))
                .keyGroup().id();
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- For API details, see CreateKeyGroup in *AWS SDK for Java 2.x API Reference*.

CreatePublicKey

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

SDK for Java 2.x

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

The following code example reads in a public key and uploads it to Amazon CloudFront.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class CreatePublicKey {
    private static final Logger logger =
 LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient, String
 publicKeyFileName) {
        try (InputStream is =
 CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                    .createPublicKey(b -> b.publicKeyConfig(c -> c
                            .name("JavaCreatedPublicKey" + UUID.randomUUID())
                            .encodedKey(publicKeyString)
                            .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
```

```
            throw new RuntimeException(e);
        }
    }
}
```

- For API details, see CreatePublicKey in *AWS SDK for Java 2.x API Reference*.

DeleteDistribution

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

SDK for Java 2.x

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

The following code example updates a distribution to *disabled*, uses a waiter that waits for the change to be deployed, then deletes the distribution.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

public class DeleteDistribution {
        private static final Logger logger =
 LoggerFactory.getLogger(DeleteDistribution.class);

        public static void deleteDistribution(final CloudFrontClient
 cloudFrontClient, final String distributionId) {
                // First, disable the distribution by updating it.
                GetDistributionResponse response =
 cloudFrontClient.getDistribution(b -> b
                                .id(distributionId));
```

```
                String etag = response.eTag();
                DistributionConfig distConfig =
response.distribution().distributionConfig();

                cloudFrontClient.updateDistribution(builder -> builder
                                .id(distributionId)
                                .distributionConfig(builder1 -> builder1

.cacheBehaviors(distConfig.cacheBehaviors())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                                .enabled(false)
                                                .origins(distConfig.origins())
                                                .comment(distConfig.comment())

.callerReference(distConfig.callerReference())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                                .priceClass(distConfig.priceClass())
                                                .aliases(distConfig.aliases())
                                                .logging(distConfig.logging())

.defaultRootObject(distConfig.defaultRootObject())

.customErrorResponses(distConfig.customErrorResponses())

.httpVersion(distConfig.httpVersion())

.isIPV6Enabled(distConfig.isIPV6Enabled())

.restrictions(distConfig.restrictions())

.viewerCertificate(distConfig.viewerCertificate())
                                                .webACLId(distConfig.webACLId())

.originGroups(distConfig.originGroups()))
                                .ifMatch(etag));

                logger.info("Distribution [{}] is DISABLED, waiting for deployment
before deleting ...",
                                distributionId);
                GetDistributionResponse distributionResponse;
                try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
```

```
                         ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                         .waitUntilDistributionDeployed(builder ->
 builder.id(distributionId)).matched();
                         distributionResponse = responseOrException.response()
                                         .orElseThrow(() -> new
 RuntimeException("Could not disable distribution"));
                 }

                 DeleteDistributionResponse deleteDistributionResponse =
 cloudFrontClient
                                 .deleteDistribution(builder -> builder
                                                 .id(distributionId)

 .ifMatch(distributionResponse.eTag()));
                 if (deleteDistributionResponse.sdkHttpResponse().isSuccessful()) {
                         logger.info("Distribution [{}] DELETED", distributionId);
                 }
         }
}
```

- For API details, see DeleteDistribution in *AWS SDK for Java 2.x API Reference*.

`UpdateDistribution`

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

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
```

```
import software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDistribution {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <id>\s

                Where:
                    id - the id value of the distribution.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String id = args[0];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        modDistribution(cloudFrontClient, id);
        cloudFrontClient.close();
    }

    public static void modDistribution(CloudFrontClient cloudFrontClient, String
 idVal) {
        try {
            // Get the Distribution to modify.
            GetDistributionRequest disRequest = GetDistributionRequest.builder()
                    .id(idVal)
                    .build();
```

```
            GetDistributionResponse response =
 cloudFrontClient.getDistribution(disRequest);
            Distribution disObject = response.distribution();
            DistributionConfig config = disObject.distributionConfig();

            // Create a new DistributionConfig object and add new values to comment
 and
            // aliases
            DistributionConfig config1 = DistributionConfig.builder()
                    .aliases(config.aliases()) // You can pass in new values here
                    .comment("New Comment")
                    .cacheBehaviors(config.cacheBehaviors())
                    .priceClass(config.priceClass())
                    .defaultCacheBehavior(config.defaultCacheBehavior())
                    .enabled(config.enabled())
                    .callerReference(config.callerReference())
                    .logging(config.logging())
                    .originGroups(config.originGroups())
                    .origins(config.origins())
                    .restrictions(config.restrictions())
                    .defaultRootObject(config.defaultRootObject())
                    .webACLId(config.webACLId())
                    .httpVersion(config.httpVersion())
                    .viewerCertificate(config.viewerCertificate())
                    .customErrorResponses(config.customErrorResponses())
                    .build();

            UpdateDistributionRequest updateDistributionRequest =
 UpdateDistributionRequest.builder()
                    .distributionConfig(config1)
                    .id(disObject.id())
                    .ifMatch(response.eTag())
                    .build();

            cloudFrontClient.updateDistribution(updateDistributionRequest);

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

- For API details, see UpdateDistribution in *AWS SDK for Java 2.x API Reference*.

Scenarios

Delete signing resources

The following code example shows how to delete resources that are used to gain access to restricted content in an Amazon Simple Storage Service (Amazon S3) bucket.

SDK for Java 2.x

Note

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

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

public class DeleteSigningResources {
    private static final Logger logger =
 LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
 cloudFrontClient,
            final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
                .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
 cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
```

```
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Origin Access Control [{}]",
 originAccessControlId);
        }
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient, final
 String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
 b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
 cloudFrontClient.deleteKeyGroup(builder -> builder
                .id(keyGroupId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
 final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
 b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
 cloudFrontClient.deletePublicKey(builder -> builder
                .id(publicKeyId)
                .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - DeleteKeyGroup
  - DeleteOriginAccessControl
  - DeletePublicKey

Sign URLs and cookies

The following code example shows how to create signed URLs and cookies that allow access to restricted resources.

SDK for Java 2.x

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

Use the CannedSignerRequest class to sign URLs or cookies with a *canned* policy.

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expirationDate)
                .build();
```

```
    }
}
```

Use the CustomSignerRequest class to sign URLs or cookies with a *custom* policy. The `activeDate` and `ipRange` are optional methods.

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expireDate)
                .activeDate(activeDate) // Optional.
                // .ipRange("192.168.0.1/24") // Optional.
                .build();
    }
}
```

The following example demonstrates the use of the CloudFrontUtilities class to produce signed cookies and URLs. View this code example on GitHub.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
 LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
 CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
 cannedSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
 customSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy
 getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
                .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
 cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCannedPolicy.keyPairIdHeaderValue());
```

```
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }

    public static CookiesForCustomPolicy
 getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
                .getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
 cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCustomPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}
```

- For API details, see CloudFrontUtilities in *AWS SDK for Java 2.x API Reference*.

CloudWatch examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudWatch.

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

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

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

Get started

Hello CloudWatch

The following code examples show how to get started using CloudWatch.

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
```

```
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> System.out.println(" Retrieved metric is: "
 + metrics.metricName()));

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

- For API details, see ListMetrics in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

DeleteAlarms

The following code example shows how to use DeleteAlarms.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/cloudwatch#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteAlarm {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to delete (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_2;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
```

```
                .build();

        deleteCWAlarm(cw, alarmName);
        cw.close();
    }

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

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

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

- For API details, see DeleteAlarms in *AWS SDK for Java 2.x API Reference*.

DeleteAnomalyDetector

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

SDK for Java 2.x

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

```
    public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
```

```
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.deleteAnomalyDetector(request);
            System.out.println("Successfully deleted the Anomaly Detector.");

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

- For API details, see DeleteAnomalyDetector in *AWS SDK for Java 2.x API Reference*.

DeleteDashboards

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

SDK for Java 2.x

Note

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

```
    public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
        try {
            DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
                    .dashboardNames(dashboardName)
                    .build();
            cw.deleteDashboards(dashboardsRequest);
            System.out.println(dashboardName + " was successfully deleted.");

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

- For API details, see DeleteDashboards in *AWS SDK for Java 2.x API Reference*.

`DescribeAlarmHistory`

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

SDK for Java 2.x

Note

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

```
    public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
```

```
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String alarmName = rootNode.findValue("exampleAlarmName").asText();

            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();
            DescribeAlarmHistoryRequest historyRequest =
 DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

            DescribeAlarmHistoryResponse response =
 cw.describeAlarmHistory(historyRequest);
            List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
            if (historyItems.isEmpty()) {
                System.out.println("No alarm history data found for " + alarmName +
 ".");
            } else {
                for (AlarmHistoryItem item : historyItems) {
                    System.out.println("History summary: " + item.historySummary());
                    System.out.println("Time stamp: " + item.timestamp());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeAlarmHistory in *AWS SDK for Java 2.x API Reference*.

DescribeAlarms

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

SDK for Java 2.x

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

```
    public static void describeAlarms(CloudWatchClient cw) {
        try {
            List<AlarmType> typeList = new ArrayList<>();
            typeList.add(AlarmType.METRIC_ALARM);

            DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
                    .alarmTypes(typeList)
                    .maxRecords(10)
                    .build();

            DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
            List<MetricAlarm> alarmList = response.metricAlarms();
            for (MetricAlarm alarm : alarmList) {
                System.out.println("Alarm name: " + alarm.alarmName());
                System.out.println("Alarm description: " +
 alarm.alarmDescription());
            }
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeAlarms in *AWS SDK for Java 2.x API Reference*.

`DescribeAlarmsForMetric`

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/cloudwatch#code-examples).

```
    public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
 rootNode.findValue("customMetricName").asText();
            boolean hasAlarm = false;
            int retries = 10;

            DescribeAlarmsForMetricRequest metricRequest =
 DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            while (!hasAlarm && retries > 0) {
                DescribeAlarmsForMetricResponse response =
 cw.describeAlarmsForMetric(metricRequest);
                hasAlarm = response.hasMetricAlarms();
                retries--;
                Thread.sleep(20000);
                System.out.println(".");
            }
            if (!hasAlarm)
                System.out.println("No Alarm state found for " + customMetricName +
 " after 10 retries.");
            else
                System.out.println("Alarm state found for " + customMetricName +
 ".");
```

```
        } catch (CloudWatchException | IOException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeAlarmsForMetric in *AWS SDK for Java 2.x API Reference*.

DescribeAnomalyDetectors

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

SDK for Java 2.x

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

```
    public static void describeAnomalyDetectors(CloudWatchClient cw, String
 fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
 rootNode.findValue("customMetricName").asText();
            DescribeAnomalyDetectorsRequest detectorsRequest =
 DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            DescribeAnomalyDetectorsResponse response =
 cw.describeAnomalyDetectors(detectorsRequest);
            List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
```

```
            for (AnomalyDetector detector : anomalyDetectorList) {
                System.out.println("Metric name: " +
 detector.singleMetricAnomalyDetector().metricName());
                System.out.println("State: " + detector.stateValue());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeAnomalyDetectors in *AWS SDK for Java 2.x API Reference*.

DisableAlarmActions

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DisableAlarmActions {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to disable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        disableActions(cw, alarmName);
        cw.close();
    }

    public static void disableActions(CloudWatchClient cw, String alarmName) {
        try {
            DisableAlarmActionsRequest request =
 DisableAlarmActionsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.disableAlarmActions(request);
            System.out.printf("Successfully disabled actions on alarm %s",
 alarmName);

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

- For API details, see DisableAlarmActions in *AWS SDK for Java 2.x API Reference*.

EnableAlarmActions

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to enable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarm = args[0];
        Region region = Region.US_EAST_1;
```

```
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        enableActions(cw, alarm);
        cw.close();
    }

    public static void enableActions(CloudWatchClient cw, String alarm) {
        try {
            EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
                    .alarmNames(alarm)
                    .build();

            cw.enableAlarmActions(request);
            System.out.printf("Successfully enabled actions on alarm %s", alarm);

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

- For API details, see EnableAlarmActions in *AWS SDK for Java 2.x API Reference*.

`GetMetricData`

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

SDK for Java 2.x

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

```
    public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
```

```
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
 rootNode.findValue("customMetricName").asText();

            // Set the date.
            Instant nowDate = Instant.now();

            long hours = 1;
            long minutes = 30;
            Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
                    ChronoUnit.MINUTES);

            Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(1)
                    .metric(met)
                    .build();

            MetricDataQuery dataQUery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

            List<MetricDataQuery> dq = new ArrayList<>();
            dq.add(dataQUery);

            GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(date2)
                    .metricDataQueries(dq)
                    .build();
```

```
            GetMetricDataResponse response = cw.getMetricData(getMetReq);
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                System.out.println("The label is " + item.label());
                System.out.println("The status code is " +
item.statusCode().toString());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see GetMetricData in *AWS SDK for Java 2.x API Reference*.

`GetMetricStatistics`

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

SDK for Java 2.x

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

```
    public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
            String metricOption, String date, Dimension myDimension) {
        try {
            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();

            GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
                    .endTime(endDate)
                    .startTime(start)
                    .dimensions(myDimension)
                    .metricName(metVal)
```

```
                    .namespace(nameSpace)
                    .period(86400)
                    .statistics(Statistic.fromValue(metricOption))
                    .build();

            GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
                }
            } else {
                System.out.println("The returned data list is empty");
            }

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

- For API details, see GetMetricStatistics in *AWS SDK for Java 2.x API Reference*.

`GetMetricWidgetImage`

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

SDK for Java 2.x

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

```
    public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
        System.out.println("Getting Image data for custom metric.");
        try {
```

```
            String myJSON = "{\n" +
                    "  \"title\": \"Example Metric Graph\",\n" +
                    "  \"view\": \"timeSeries\",\n" +
                    "  \"stacked \": false,\n" +
                    "  \"period\": 10,\n" +
                    "  \"width\": 1400,\n" +
                    "  \"height\": 600,\n" +
                    "  \"metrics\": [\n" +
                    "    [\n" +
                    "      \"AWS/Billing\",\n" +
                    "      \"EstimatedCharges\",\n" +
                    "      \"Currency\",\n" +
                    "      \"USD\"\n" +
                    "    ]\n" +
                    "  ]\n" +
                    "}";

            GetMetricWidgetImageRequest imageRequest =
 GetMetricWidgetImageRequest.builder()
                    .metricWidget(myJSON)
                    .build();

            GetMetricWidgetImageResponse response =
 cw.getMetricWidgetImage(imageRequest);
            SdkBytes sdkBytes = response.metricWidgetImage();
            byte[] bytes = sdkBytes.asByteArray();
            File outputFile = new File(fileName);
            try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
                outputStream.write(bytes);
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see GetMetricWidgetImage in *AWS SDK for Java 2.x API Reference*.

ListDashboards

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/cloudwatch#code-examples).

```
    public static void listDashboards(CloudWatchClient cw) {
        try {
            ListDashboardsIterable listRes = cw.listDashboardsPaginator();
            listRes.stream()
                    .flatMap(r -> r.dashboardEntries().stream())
                    .forEach(entry -> {
                        System.out.println("Dashboard name is: " +
 entry.dashboardName());
                        System.out.println("Dashboard ARN is: " +
 entry.dashboardArn());
                    });

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

- For API details, see ListDashboards in *AWS SDK for Java 2.x API Reference*.

`ListMetrics`

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/cloudwatch#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMetrics {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
```

```
        boolean done = false;
        String nextToken = null;

        try {
            while (!done) {

                ListMetricsResponse response;
                if (nextToken == null) {
                    ListMetricsRequest request = ListMetricsRequest.builder()
                            .namespace(namespace)
                            .build();

                    response = cw.listMetrics(request);
                } else {
                    ListMetricsRequest request = ListMetricsRequest.builder()
                            .namespace(namespace)
                            .nextToken(nextToken)
                            .build();

                    response = cw.listMetrics(request);
                }

                for (Metric metric : response.metrics()) {
                    System.out.printf("Retrieved metric %s", metric.metricName());
                    System.out.println();
                }

                if (response.nextToken() == null) {
                    done = true;
                } else {
                    nextToken = response.nextToken();
                }
            }

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

- For API details, see ListMetrics in *AWS SDK for Java 2.x API Reference*.

PutAnomalyDetector

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

SDK for Java 2.x

Note

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

```
    public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
 rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
 SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            PutAnomalyDetectorRequest anomalyDetectorRequest =
 PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.putAnomalyDetector(anomalyDetectorRequest);
            System.out.println("Added anomaly detector for metric " +
 customMetricName + ".");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
```

- For API details, see PutAnomalyDetector in *AWS SDK for Java 2.x API Reference*.

PutDashboard

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

SDK for Java 2.x

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

```
    public static void createDashboardWithMetrics(CloudWatchClient cw, String
 dashboardName, String fileName) {
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                    .dashboardName(dashboardName)
                    .dashboardBody(readFileAsString(fileName))
                    .build();

            PutDashboardResponse response = cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName + " was successfully created.");
            List<DashboardValidationMessage> messages =
 response.dashboardValidationMessages();
            if (messages.isEmpty()) {
                System.out.println("There are no messages in the new Dashboard");
            } else {
                for (DashboardValidationMessage message : messages) {
                    System.out.println("Message is: " + message.message());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see PutDashboard in *AWS SDK for Java 2.x API Reference*.

PutMetricAlarm

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

SDK for Java 2.x

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

```
    public static String createAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
 rootNode.findValue("customMetricName").asText();
            String alarmName = rootNode.findValue("exampleAlarmName").asText();
            String emailTopic = rootNode.findValue("emailTopic").asText();
            String accountId = rootNode.findValue("accountId").asText();
            String region = rootNode.findValue("region").asText();

            // Create a List for alarm actions.
            List<String> alarmActions = new ArrayList<>();
            alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
 emailTopic);
            PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
                    .alarmActions(alarmActions)
                    .alarmDescription("Example metric alarm")
                    .alarmName(alarmName)

 .comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
                    .threshold(100.00)
```

```
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .evaluationPeriods(1)
                    .period(10)
                    .statistic("Maximum")
                    .datapointsToAlarm(1)
                    .treatMissingData("ignore")
                    .build();

            cw.putMetricAlarm(alarmRequest);
            System.out.println(alarmName + " was successfully created!");
            return alarmName;

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see PutMetricAlarm in *AWS SDK for Java 2.x API Reference*.

PutMetricData

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

SDK for Java 2.x

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

```
    public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
```

```
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            // Set an Instant object.
            String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

            MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

            List<MetricDatum> metricDataList = new ArrayList<>();
            metricDataList.add(datum);
            metricDataList.add(datum2);

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric " +
customMetricName);

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see PutMetricData in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with metrics, dashboards, and alarms

The following code example shows how to:

- List CloudWatch namespaces and metrics.
- Get statistics for a metric and for estimated billing.
- Create and update a dashboard.
- Create and add data to a metric.
- Create and trigger an alarm, then view alarm history.
- Add an anomaly detector.
- Get a metric image, then clean up resources.

SDK for Java 2.x

Note

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

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.AlarmHistoryItem;
import software.amazon.awssdk.services.cloudwatch.model.AlarmType;
import software.amazon.awssdk.services.cloudwatch.model.AnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.DashboardValidationMessage;
import software.amazon.awssdk.services.cloudwatch.model.Datapoint;
```

```
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricResponse;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageResponse;
import software.amazon.awssdk.services.cloudwatch.model.HistoryItemType;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataQuery;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataResult;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.MetricStat;
import software.amazon.awssdk.services.cloudwatch.model.PutAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.ScanBy;
import software.amazon.awssdk.services.cloudwatch.model.SingleMetricAnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.paginators.ListDashboardsIterable;
```

```
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
```

```
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                  <myDate> <costDateWeek> <dashboardName> <dashboardJson>
 <dashboardAdd> <settings> <metricImage> \s

                Where:
                  myDate - The start date to use to get metric statistics. (For
 example, 2023-01-11T18:35:24.00Z.)\s
                  costDateWeek - The start date to use to get AWS/Billinget
 statistics. (For example, 2023-01-11T18:35:24.00Z.)\s
                  dashboardName - The name of the dashboard to create.\s
                  dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)\s
                  dashboardAdd - The location of a JSON file to use to update a
 dashboard. (See Readme file.)\s
                  settings - The location of a JSON file from which various values
 are read. (See Readme file.)\s
                  metricImage - The location of a BMP file that is used to create a
 graph.\s
                """;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        Region region = Region.US_EAST_1;
        String myDate = args[0];
        String costDateWeek = args[1];
        String dashboardName = args[2];
        String dashboardJson = args[3];
        String dashboardAdd = args[4];
```

```
        String settings = args[5];
        String metricImage = args[6];

        Double dataPoint = Double.parseDouble("10.0");
        Scanner sc = new Scanner(System.in);
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon CloudWatch example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "1. List at least five available unique namespaces from Amazon
 CloudWatch. Select one from the list.");
        ArrayList<String> list = listNameSpaces(cw);
        for (int z = 0; z < 5; z++) {
            int index = z + 1;
            System.out.println("    " + index + ". " + list.get(z));
        }

        String selectedNamespace = "";
        String selectedMetrics = "";
        int num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            selectedNamespace = list.get(num - 1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + selectedNamespace);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List available metrics within the selected namespace
 and select one from the list.");
        ArrayList<String> metList = listMets(cw, selectedNamespace);
        for (int z = 0; z < 5; z++) {
            int index = z + 1;
            System.out.println("    " + index + ". " + metList.get(z));
        }
```

```
        num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            selectedMetrics = metList.get(num - 1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + selectedMetrics);
        Dimension myDimension = getSpecificMet(cw, selectedNamespace);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get statistics for the selected metric over the last
 day.");
        String metricOption = "";
        ArrayList<String> statTypes = new ArrayList<>();
        statTypes.add("SampleCount");
        statTypes.add("Average");
        statTypes.add("Sum");
        statTypes.add("Minimum");
        statTypes.add("Maximum");

        for (int t = 0; t < 5; t++) {
            System.out.println("    " + (t + 1) + ". " + statTypes.get(t));
        }
        System.out.println("Select a metric statistic by entering a number from the
 preceding list:");
        num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            metricOption = statTypes.get(num - 1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + metricOption);
        getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
 metricOption, myDate, myDimension);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get CloudWatch estimated billing for the last
 week.");
        getMetricStatistics(cw, costDateWeek);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("5. Create a new CloudWatch dashboard with metrics.");
        createDashboardWithMetrics(cw, dashboardName, dashboardJson);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List dashboards using a paginator.");
        listDashboards(cw);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a new custom metric by adding data to it.");
        createNewCustomMetric(cw, dataPoint);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Add an additional metric to the dashboard.");
        addMetricToDashboard(cw, dashboardAdd, dashboardName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Create an alarm for the custom metric.");
        String alarmName = createAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Describe ten current alarms.");
        describeAlarms(cw);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Get current data for new custom metric.");
        getCustomMetricData(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Push data into the custom metric to trigger the
 alarm.");
        addMetricDataForAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("13. Check the alarm state using the action
 DescribeAlarmsForMetric.");
        checkForMetricAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Get alarm history for the new alarm.");
        getAlarmHistory(cw, settings, myDate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Add an anomaly detector for the custom metric.");
        addAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Describe current anomaly detectors.");
        describeAnomalyDetectors(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Get a metric image for the custom metric.");
        getAndOpenMetricImage(cw, metricImage);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up the Amazon CloudWatch resources.");
        deleteDashboard(cw, dashboardName);
        deleteCWAlarm(cw, alarmName);
        deleteAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon CloudWatch example scenario is complete.");
        System.out.println(DASHES);
        cw.close();
    }

    public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
```

```
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.deleteAnomalyDetector(request);
            System.out.println("Successfully deleted the Anomaly Detector.");

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

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.deleteAlarms(request);
            System.out.println("Successfully deleted alarm " + alarmName);

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

    public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
```

```
        try {
            DeleteDashboardsRequest dashboardsRequest =
 DeleteDashboardsRequest.builder()
                    .dashboardNames(dashboardName)
                    .build();
            cw.deleteDashboards(dashboardsRequest);
            System.out.println(dashboardName + " was successfully deleted.");

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
        System.out.println("Getting Image data for custom metric.");
        try {
            String myJSON = "{\n" +
                    "  \"title\": \"Example Metric Graph\",\n" +
                    "  \"view\": \"timeSeries\",\n" +
                    "  \"stacked \": false,\n" +
                    "  \"period\": 10,\n" +
                    "  \"width\": 1400,\n" +
                    "  \"height\": 600,\n" +
                    "  \"metrics\": [\n" +
                    "    [\n" +
                    "      \"AWS/Billing\",\n" +
                    "      \"EstimatedCharges\",\n" +
                    "      \"Currency\",\n" +
                    "      \"USD\"\n" +
                    "    ]\n" +
                    "  ]\n" +
                    "}";

            GetMetricWidgetImageRequest imageRequest =
 GetMetricWidgetImageRequest.builder()
                    .metricWidget(myJSON)
                    .build();

            GetMetricWidgetImageResponse response =
 cw.getMetricWidgetImage(imageRequest);
            SdkBytes sdkBytes = response.metricWidgetImage();
            byte[] bytes = sdkBytes.asByteArray();
            File outputFile = new File(fileName);
```

```
            try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
                outputStream.write(bytes);
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void describeAnomalyDetectors(CloudWatchClient cw, String
 fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
 rootNode.findValue("customMetricName").asText();
            DescribeAnomalyDetectorsRequest detectorsRequest =
 DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            DescribeAnomalyDetectorsResponse response =
 cw.describeAnomalyDetectors(detectorsRequest);
            List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
            for (AnomalyDetector detector : anomalyDetectorList) {
                System.out.println("Metric name: " +
 detector.singleMetricAnomalyDetector().metricName());
                System.out.println("State: " + detector.stateValue());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
```

```
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
 rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
 SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            PutAnomalyDetectorRequest anomalyDetectorRequest =
 PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.putAnomalyDetector(anomalyDetectorRequest);
            System.out.println("Added anomaly detector for metric " +
 customMetricName + ".");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
 date) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String alarmName = rootNode.findValue("exampleAlarmName").asText();

            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();
```

```
            DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

            DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
            List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
            if (historyItems.isEmpty()) {
                System.out.println("No alarm history data found for " + alarmName +
".");
            } else {
                for (AlarmHistoryItem item : historyItems) {
                    System.out.println("History summary: " + item.historySummary());
                    System.out.println("Time stamp: " + item.timestamp());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            boolean hasAlarm = false;
            int retries = 10;

            DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
```

```
                    .build();

            while (!hasAlarm && retries > 0) {
                DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
                hasAlarm = response.hasMetricAlarms();
                retries--;
                Thread.sleep(20000);
                System.out.println(".");
            }
            if (!hasAlarm)
                System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
            else
                System.out.println("Alarm state found for " + customMetricName +
".");

        } catch (CloudWatchException | IOException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            // Set an Instant object.
            String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
```

```
                    .build();

            MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

            List<MetricDatum> metricDataList = new ArrayList<>();
            metricDataList.add(datum);
            metricDataList.add(datum2);

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric " +
 customMetricName);

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
 rootNode.findValue("customMetricName").asText();

            // Set the date.
            Instant nowDate = Instant.now();

            long hours = 1;
            long minutes = 30;
```

```
            Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
                    ChronoUnit.MINUTES);

            Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(1)
                    .metric(met)
                    .build();

            MetricDataQuery dataQUery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

            List<MetricDataQuery> dq = new ArrayList<>();
            dq.add(dataQUery);

            GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(date2)
                    .metricDataQueries(dq)
                    .build();

            GetMetricDataResponse response = cw.getMetricData(getMetReq);
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                System.out.println("The label is " + item.label());
                System.out.println("The status code is " +
 item.statusCode().toString());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
    public static void describeAlarms(CloudWatchClient cw) {
        try {
            List<AlarmType> typeList = new ArrayList<>();
            typeList.add(AlarmType.METRIC_ALARM);

            DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
                    .alarmTypes(typeList)
                    .maxRecords(10)
                    .build();

            DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
            List<MetricAlarm> alarmList = response.metricAlarms();
            for (MetricAlarm alarm : alarmList) {
                System.out.println("Alarm name: " + alarm.alarmName());
                System.out.println("Alarm description: " +
 alarm.alarmDescription());
            }
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
 rootNode.findValue("customMetricName").asText();
            String alarmName = rootNode.findValue("exampleAlarmName").asText();
            String emailTopic = rootNode.findValue("emailTopic").asText();
            String accountId = rootNode.findValue("accountId").asText();
            String region = rootNode.findValue("region").asText();

            // Create a List for alarm actions.
            List<String> alarmActions = new ArrayList<>();
            alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
 emailTopic);
            PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
```

```
                    .alarmActions(alarmActions)
                    .alarmDescription("Example metric alarm")
                    .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
                    .threshold(100.00)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .evaluationPeriods(1)
                    .period(10)
                    .statistic("Maximum")
                    .datapointsToAlarm(1)
                    .treatMissingData("ignore")
                    .build();

            cw.putMetricAlarm(alarmRequest);
            System.out.println(alarmName + " was successfully created!");
            return alarmName;

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                    .dashboardName(dashboardName)
                    .dashboardBody(readFileAsString(fileName))
                    .build();

            cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName + " was successfully updated.");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
    public static void createNewCustomMetric(CloudWatchClient cw, Double dataPoint)
 {
        try {
            Dimension dimension = Dimension.builder()
                    .name("UNIQUE_PAGES")
                    .value("URLS")
                    .build();

            // Set an Instant object.
            String time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                    .metricName("PAGES_VISITED")
                    .unit(StandardUnit.NONE)
                    .value(dataPoint)
                    .timestamp(instant)
                    .dimensions(dimension)
                    .build();

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace("SITE/TRAFFIC")
                    .metricData(datum)
                    .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric PAGES_VISITED");

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

    public static void listDashboards(CloudWatchClient cw) {
        try {
            ListDashboardsIterable listRes = cw.listDashboardsPaginator();
            listRes.stream()
                    .flatMap(r -> r.dashboardEntries().stream())
                    .forEach(entry -> {
                        System.out.println("Dashboard name is: " +
 entry.dashboardName());
```

```
                        System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
                    });

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

    public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                    .dashboardName(dashboardName)
                    .dashboardBody(readFileAsString(fileName))
                    .build();

            PutDashboardResponse response = cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName + " was successfully created.");
            List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
            if (messages.isEmpty()) {
                System.out.println("There are no messages in the new Dashboard");
            } else {
                for (DashboardValidationMessage message : messages) {
                    System.out.println("Message is: " + message.message());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }

    public static void getMetricStatistics(CloudWatchClient cw, String costDateWeek)
{
        try {
            Instant start = Instant.parse(costDateWeek);
```

```
            Instant endDate = Instant.now();
            Dimension dimension = Dimension.builder()
                    .name("Currency")
                    .value("USD")
                    .build();

            List<Dimension> dimensionList = new ArrayList<>();
            dimensionList.add(dimension);
            GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
                    .metricName("EstimatedCharges")
                    .namespace("AWS/Billing")
                    .dimensions(dimensionList)
                    .statistics(Statistic.MAXIMUM)
                    .startTime(start)
                    .endTime(endDate)
                    .period(86400)
                    .build();

            GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
                }
            } else {
                System.out.println("The returned data list is empty");
            }

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

    public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
            String metricOption, String date, Dimension myDimension) {
        try {
            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();
```

```
            GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
                    .endTime(endDate)
                    .startTime(start)
                    .dimensions(myDimension)
                    .metricName(metVal)
                    .namespace(nameSpace)
                    .period(86400)
                    .statistics(Statistic.fromValue(metricOption))
                    .build();

            GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
                }
            } else {
                System.out.println("The returned data list is empty");
            }

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static Dimension getSpecificMet(CloudWatchClient cw, String namespace) {
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsResponse response = cw.listMetrics(request);
            List<Metric> myList = response.metrics();
            Metric metric = myList.get(0);
            return metric.dimensions().get(0);

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

```
            System.exit(1);
        }
        return null;
    }

    public static ArrayList<String> listMets(CloudWatchClient cw, String namespace)
 {
        try {
            ArrayList<String> metList = new ArrayList<>();
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> metList.add(metrics.metricName()));

            return metList;

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

    public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
        try {
            ArrayList<String> nameSpaceList = new ArrayList<>();
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> {
                        String data = metrics.namespace();
                        if (!nameSpaceList.contains(data)) {
                            nameSpaceList.add(data);
                        }
                    });

            return nameSpaceList;
```

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

CloudWatch Events examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudWatch Events.

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

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

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

PutEvents

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutEvents {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <resourceArn>

                Where:
                    resourceArn - An Amazon Resource Name (ARN) related to the
 events.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWEvents(cwe, resourceArn);
        cwe.close();
    }

    public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn) {
        try {
            final String EVENT_DETAILS = "{ \"key1\": \"value1\", \"key2\":
 \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
                    .detail(EVENT_DETAILS)
                    .detailType("sampleSubmitted")
                    .resources(resourceArn)
                    .source("aws-sdk-java-cloudwatch-example")
                    .build();

            PutEventsRequest request = PutEventsRequest.builder()
                    .entries(requestEntry)
                    .build();

            cwe.putEvents(request);
            System.out.println("Successfully put CloudWatch event");

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

```
}
```

- For API details, see PutEvents in *AWS SDK for Java 2.x API Reference*.

PutRule

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <ruleName> roleArn>\s

                Where:
                  ruleName - A rule name (for example, myrule).
                  roleArn - A role ARN value (for example,
 arn:aws:iam::xxxxxx047983:user/MyUser).
```

```
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String roleArn = args[1];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWRule(cwe, ruleName, roleArn);
        cwe.close();
    }

    public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
 roleArn) {
        try {
            PutRuleRequest request = PutRuleRequest.builder()
                    .name(ruleName)
                    .roleArn(roleArn)
                    .scheduleExpression("rate(5 minutes)")
                    .state(RuleState.ENABLED)
                    .build();

            PutRuleResponse response = cwe.putRule(request);
            System.out.printf(
                    "Successfully created CloudWatch events rule %s with arn %s",
                    roleArn, response.ruleArn());

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

- For API details, see PutRule in *AWS SDK for Java 2.x API Reference*.

PutTargets

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

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutTargets {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <ruleName> <functionArn> <targetId>\s

                Where:
                  ruleName - A rule name (for example, myrule).
                  functionArn - An AWS Lambda function ARN (for example,
 arn:aws:lambda:us-west-2:xxxxxx047983:function:lamda1).
                  targetId - A target id value.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String ruleName = args[0];
        String functionArn = args[1];
        String targetId = args[2];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWTargets(cwe, ruleName, functionArn, targetId);
        cwe.close();
    }

    public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName,
 String functionArn, String targetId) {
        try {
            Target target = Target.builder()
                    .arn(functionArn)
                    .id(targetId)
                    .build();

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

            cwe.putTargets(request);
            System.out.printf(
                    "Successfully created CloudWatch events target for rule %s",
                    ruleName);

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

- For API details, see PutTargets in *AWS SDK for Java 2.x API Reference*.

CloudWatch Logs examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudWatch Logs.

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

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

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

DeleteSubscriptionFilter

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <filter> <logGroup>

                Where:
                  filter - The name of the subscription filter (for example,
 MyFilter).
                  logGroup - The name of the log group. (for example, testgroup).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
 String logGroup) {
        try {
            DeleteSubscriptionFilterRequest request =
 DeleteSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .logGroupName(logGroup)
                    .build();

            logs.deleteSubscriptionFilter(request);
            System.out.printf("Successfully deleted CloudWatch logs subscription
 filter %s", filter);
```

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

- For API details, see [DeleteSubscriptionFilter](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeSubscriptionFilters`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                  <logGroup>

                Where:
                  logGroup - A log group name (for example, myloggroup).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String logGroup = args[0];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        describeFilters(logs, logGroup);
        logs.close();
    }

    public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {
        try {
            boolean done = false;
            String newToken = null;

            while (!done) {
                DescribeSubscriptionFiltersResponse response;
                if (newToken == null) {
                    DescribeSubscriptionFiltersRequest request =
 DescribeSubscriptionFiltersRequest.builder()
                            .logGroupName(logGroup)
                            .limit(1).build();

                    response = logs.describeSubscriptionFilters(request);
                } else {
                    DescribeSubscriptionFiltersRequest request =
 DescribeSubscriptionFiltersRequest.builder()
                            .nextToken(newToken)
                            .logGroupName(logGroup)
                            .limit(1).build();
                    response = logs.describeSubscriptionFilters(request);
```

```
                }

                for (SubscriptionFilter filter : response.subscriptionFilters()) {
                    System.out.printf("Retrieved filter with name %s, " + "pattern
 %s " + "and destination arn %s",
                            filter.filterName(),
                            filter.filterPattern(),
                            filter.destinationArn());
                }

                if (response.nextToken() == null) {
                    done = true;
                } else {
                    newToken = response.nextToken();
                }
            }

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

- For API details, see DescribeSubscriptionFilters in *AWS SDK for Java 2.x API Reference*.

`PutSubscriptionFilter`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
```

```
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
 * '111111111111' with your account details.
 * For more information, see "Subscription Filters with AWS Lambda" in the
 * Amazon CloudWatch Logs Guide.
 *
 *
 * Also, before running this Java V2 code example,set up your development
 * environment,including your credentials.
 *
 * For more information,see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */

public class PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <filter> <pattern> <logGroup> <functionArn>\s

                Where:
                  filter - A filter name (for example, myfilter).
                  pattern - A filter pattern (for example, ERROR).
                  logGroup - A log group name (testgroup).
                  functionArn - An AWS Lambda function ARN (for example,
 arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
                """;
```

```
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String pattern = args[1];
        String logGroup = args[2];
        String functionArn = args[3];
        Region region = Region.US_WEST_2;
        CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
                .region(region)
                .build();

        putSubFilters(cwl, filter, pattern, logGroup, functionArn);
        cwl.close();
    }

    public static void putSubFilters(CloudWatchLogsClient cwl,
            String filter,
            String pattern,
            String logGroup,
            String functionArn) {

        try {
            PutSubscriptionFilterRequest request =
 PutSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .filterPattern(pattern)
                    .logGroupName(logGroup)
                    .destinationArn(functionArn)
                    .build();

            cwl.putSubscriptionFilter(request);
            System.out.printf(
                    "Successfully created CloudWatch logs subscription filter %s",
                    filter);

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

- For API details, see PutSubscriptionFilter in *AWS SDK for Java 2.x API Reference*.

StartLiveTail

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

SDK for Java 2.x

Include the required files.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Handle the events from the Live Tail session.

```
    private static StartLiveTailResponseHandler
 getStartLiveTailResponseStreamHandler(
            AtomicReference<Subscription> subscriptionAtomicReference) {
        return StartLiveTailResponseHandler.builder()
            .onResponse(r -> System.out.println("Received initial response"))
            .onError(throwable -> {
                CloudWatchLogsException e = (CloudWatchLogsException)
 throwable.getCause();
                System.err.println(e.awsErrorDetails().errorMessage());
```

```
                System.exit(1);
            })
            .subscriber(() -> new FlowableSubscriber<>() {
                @Override
                public void onSubscribe(@NonNull Subscription s) {
                    subscriptionAtomicReference.set(s);
                    s.request(Long.MAX_VALUE);
                }

                @Override
                public void onNext(StartLiveTailResponseStream event) {
                    if (event instanceof LiveTailSessionStart) {
                        LiveTailSessionStart sessionStart = (LiveTailSessionStart)
 event;
                        System.out.println(sessionStart);
                    } else if (event instanceof LiveTailSessionUpdate) {
                        LiveTailSessionUpdate sessionUpdate =
 (LiveTailSessionUpdate) event;
                        List<LiveTailSessionLogEvent> logEvents =
 sessionUpdate.sessionResults();
                        logEvents.forEach(e -> {
                            long timestamp = e.timestamp();
                            Date date = new Date(timestamp);
                            System.out.println("[" + date + "] " + e.message());
                        });
                    } else {
                        throw CloudWatchLogsException.builder().message("Unknown
 event type").build();
                    }
                }

                @Override
                public void onError(Throwable throwable) {
                    System.out.println(throwable.getMessage());
                    System.exit(1);
                }

                @Override
                public void onComplete() {
                    System.out.println("Completed Streaming Session");
                }
            })
            .build();
    }
```

Start the Live Tail session.

```
        CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
                CloudWatchLogsAsyncClient.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();

        StartLiveTailRequest request =
                StartLiveTailRequest.builder()
                    .logGroupIdentifiers(logGroupIdentifiers)
                    .logStreamNames(logStreamNames)
                    .logEventFilterPattern(logEventFilterPattern)
                    .build();

        /* Create a reference to store the subscription */
        final AtomicReference<Subscription> subscriptionAtomicReference = new
 AtomicReference<>(null);

        cloudWatchLogsAsyncClient.startLiveTail(request,
 getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));
```

Stop the Live Tail session after a period of time has elapsed.

```
        /* Set a timeout for the session and cancel the subscription. This will:
         * 1). Close the stream
         * 2). Stop the Live Tail session
         */
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        if (subscriptionAtomicReference.get() != null) {
            subscriptionAtomicReference.get().cancel();
            System.out.println("Subscription to stream closed");
        }
```

- For API details, see StartLiveTail in *AWS SDK for Java 2.x API Reference*.

Amazon Cognito Identity examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Cognito Identity.

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

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

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

CreateIdentityPool

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <identityPoolName>\s

                Where:
                    identityPoolName - The name to give your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
 identityPoolName) {
        try {
            CreateIdentityPoolRequest poolRequest =
 CreateIdentityPoolRequest.builder()
                    .allowUnauthenticatedIdentities(false)
                    .identityPoolName(identityPoolName)
                    .build();

            CreateIdentityPoolResponse response =
 cognitoClient.createIdentityPool(poolRequest);
```

```
            return response.identityPoolId();

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

- For API details, see CreateIdentityPool in *AWS SDK for Java 2.x API Reference*.

DeleteIdentityPool

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteIdentityPool {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <identityPoolId>\s

                Where:
                    identityPoolId - The Id value of your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoold = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        deleteIdPool(cognitoIdClient, identityPoold);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
 identityPoold) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
 DeleteIdentityPoolRequest.builder()
                    .identityPoolId(identityPoold)
                    .build();

            cognitoIdClient.deleteIdentityPool(identityPoolRequest);
            System.out.println("Done");

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

- For API details, see [DeleteIdentityPool](#) in *AWS SDK for Java 2.x API Reference*.

GetCredentialsForIdentity

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s
```

```
            Where:
                identityId - The Id of an existing identity in the format
 REGION:GUID.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
 String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
 GetCredentialsForIdentityRequest
                    .builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                    .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                    "Identity ID " + response.identityId() + ", Access key ID " +
 response.credentials().accessKeyId());

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

- For API details, see GetCredentialsForIdentity in *AWS SDK for Java 2.x API Reference*.

ListIdentityPools

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

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
```

```
                    .maxResults(15)
                    .build();

            ListIdentityPoolsResponse response =
 cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

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

- For API details, see ListIdentityPools in *AWS SDK for Java 2.x API Reference*.

Amazon Cognito Identity Provider examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Cognito Identity Provider.

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

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

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

Get started

Hello Amazon Cognito

The following code examples show how to get started using Amazon Cognito.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/cognito#code-examples).

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
 {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
```

```
                    .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID " +
 userpool.id());
            });

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

- For API details, see ListUserPools in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

AdminGetUser

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

SDK for Java 2.x

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

```
    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
```

```
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

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

- For API details, see AdminGetUser in *AWS SDK for Java 2.x API Reference*.

`AdminInitiateAuth`

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

SDK for Java 2.x

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

```
    public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
            String clientId, String userName, String password, String userPoolId) {
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

            AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
```

```
                    .clientId(clientId)
                    .userPoolId(userPoolId)
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                    .build();

            AdminInitiateAuthResponse response =
 identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " + response.challengeName());
            return response;

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

        return null;
    }
```

- For API details, see AdminInitiateAuth in *AWS SDK for Java 2.x API Reference*.

AdminRespondToAuthChallenge

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

SDK for Java 2.x

Note

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

```
    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
 identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
```

```
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
        System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }
```

- For API details, see AdminRespondToAuthChallenge in *AWS SDK for Java 2.x API Reference*.

`AssociateSoftwareToken`

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

SDK for Java 2.x

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

```
    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
```

```
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }
```

- For API details, see AssociateSoftwareToken in *AWS SDK for Java 2.x API Reference*.

ConfirmSignUp

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

SDK for Java 2.x

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

```
    public static void confirmSignUp(CognitoIdentityProviderClient
 identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

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

- For API details, see ConfirmSignUp in *AWS SDK for Java 2.x API Reference*.

CreateUserPool

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

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolName>\s

                Where:
                    userPoolName - The name to give your user pool when it's
 created.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
 String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                    .poolName(userPoolName)
                    .build();

            CreateUserPoolResponse response = cognitoClient.createUserPool(request);
            return response.userPool().id();

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

- For API details, see [CreateUserPool](#) in *AWS SDK for Java 2.x API Reference*.

`CreateUserPoolClient`

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

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/cognito#code-examples).

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s

                Where:
                    clientName - The name for the user pool client to create.
```

```
                    userPoolId - The ID for the user pool.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient cognitoClient,
 String clientName,
            String userPoolId) {
        try {
            CreateUserPoolClientRequest request =
 CreateUserPoolClientRequest.builder()
                    .clientName(clientName)
                    .userPoolId(userPoolId)
                    .build();

            CreateUserPoolClientResponse response =
 cognitoClient.createUserPoolClient(request);
            System.out.println("User pool " + response.userPoolClient().clientName()
 + " created. ID: "
                    + response.userPoolClient().clientId());

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

- For API details, see CreateUserPoolClient in *AWS SDK for Java 2.x API Reference*.

ListUserPools

The following code example shows how to use ListUserPools.

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
 {
```

```
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID " +
 userpool.id());
            });

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

- For API details, see ListUserPools in *AWS SDK for Java 2.x API Reference*.

ListUsers

The following code example shows how to use ListUsers.

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolId>\s

                Where:
                    userPoolId - The ID given to your user pool when it's created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
 String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .build();
```

```
            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User " + user.username() + " Status " +
 user.userStatus() + " Created "
                        + user.userCreateDate());
            });

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

    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
 String userPoolId) {

        try {
            String filter = "email = \"tblue@noserver.com\"";
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .filter(filter)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User with filter applied " + user.username() + "
 Status " + user.userStatus()
                        + " Created " + user.userCreateDate());
            });

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

- For API details, see ListUsers in *AWS SDK for Java 2.x API Reference*.

`ResendConfirmationCode`

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

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/cognito#code-examples).

```
    public static void resendConfirmationCode(CognitoIdentityProviderClient
 identityProviderClient, String clientId,
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
 ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
 identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
 response.codeDeliveryDetails().deliveryMediumAsString());

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

- For API details, see ResendConfirmationCode in *AWS SDK for Java 2.x API Reference*.

`SignUp`

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

SDK for Java 2.x

Note

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

```
    public static void signUp(CognitoIdentityProviderClient identityProviderClient,
 String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

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

- For API details, see SignUp in *AWS SDK for Java 2.x API Reference*.

`VerifySoftwareToken`

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

SDK for Java 2.x

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

```
    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient
 identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
 VerifySoftwareTokenRequest.builder()
                    .userCode(code)
                    .session(session)
                    .build();

            VerifySoftwareTokenResponse verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is " +
 verifyResponse.statusAsString());

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

- For API details, see VerifySoftwareToken in *AWS SDK for Java 2.x API Reference*.

Scenarios

Sign up a user with a user pool that requires MFA

The following code example shows how to:

- Sign up and confirm a user with a username, password, and email address.
- Set up multi-factor authentication by associating an MFA application with the user.
- Sign in by using a password and an MFA code.

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRe
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRe
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRespons
import software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRespons
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenResponse;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {
        final String usage = """

                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the AWS
 CDK script.
                    poolId - The pool Id that you can get from the AWS CDK script.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);

        System.out
                .println("*** Conformation code sent to " + userName + ". Would you
 like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
 pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
 initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to generate
 a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient, mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
 Authenticator");
```

```
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
 Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
 initiateAuth(identityProviderClient, clientId, userName, password,
               poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
 mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
 performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
 identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
 AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
```

```
                .challengeResponses(challengeResponses)
                .session(session)
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
        System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }

    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                    .userCode(code)
                    .session(session)
                    .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is " +
verifyResponse.statusAsString());

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

    public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
            String clientId, String userName, String password, String userPoolId) {
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

            AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                    .clientId(clientId)
                    .userPoolId(userPoolId)
```

```
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                    .build();

            AdminInitiateAuthResponse response = 
identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " + response.challengeName());
            return response;

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

        return null;
    }

    public static String getSecretForAppMFA(CognitoIdentityProviderClient 
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest = 
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient 
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");
```

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

    public static void resendConfirmationCode(CognitoIdentityProviderClient
 identityProviderClient, String clientId,
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
 ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
 identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
 response.codeDeliveryDetails().deliveryMediumAsString());

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

    public static void signUp(CognitoIdentityProviderClient identityProviderClient,
 String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
```

```
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

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

    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth

- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Amazon Comprehend examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Comprehend.

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

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

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

`CreateDocumentClassifier`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
 software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-using-
amazon-comprehend/
 *
 * Also, set up your development environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <dataAccessRoleArn> <s3Uri> <documentClassifierName>

                Where:
                  dataAccessRoleArn - The ARN value of the role used for this
 operation.
                  s3Uri - The Amazon S3 bucket that contains the CSV file.
                  documentClassifierName - The name of the document classifier.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dataAccessRoleArn = args[0];
        String s3Uri = args[1];
        String documentClassifierName = args[2];
```

```
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
 documentClassifierName);
        comClient.close();
    }

    public static void createDocumentClassifier(ComprehendClient comClient, String
 dataAccessRoleArn, String s3Uri,
            String documentClassifierName) {
        try {
            DocumentClassifierInputDataConfig config =
 DocumentClassifierInputDataConfig.builder()
                    .s3Uri(s3Uri)
                    .build();

            CreateDocumentClassifierRequest createDocumentClassifierRequest =
 CreateDocumentClassifierRequest.builder()
                    .documentClassifierName(documentClassifierName)
                    .dataAccessRoleArn(dataAccessRoleArn)
                    .languageCode("en")
                    .inputDataConfig(config)
                    .build();

            CreateDocumentClassifierResponse createDocumentClassifierResult =
 comClient
                    .createDocumentClassifier(createDocumentClassifierRequest);
            String documentClassifierArn =
 createDocumentClassifierResult.documentClassifierArn();
            System.out.println("Document Classifier ARN: " + documentClassifierArn);

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

- For API details, see CreateDocumentClassifier in *AWS SDK for Java 2.x API Reference*.

DetectDominantLanguage

The following code example shows how to use DetectDominantLanguage.

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLanguage {
    public static void main(String[] args) {
        // Specify French text - "It is raining today in Seattle".
        String text = "Il pleut aujourd'hui à Seattle";
        Region region = Region.US_EAST_1;

        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectDominantLanguage");
        detectTheDominantLanguage(comClient, text);
        comClient.close();
```

```
    }

    public static void detectTheDominantLanguage(ComprehendClient comClient, String
 text) {
        try {
            DetectDominantLanguageRequest request =
 DetectDominantLanguageRequest.builder()
                    .text(text)
                    .build();

            DetectDominantLanguageResponse resp =
 comClient.detectDominantLanguage(request);
            List<DominantLanguage> allLanList = resp.languages();
            for (DominantLanguage lang : allLanList) {
                System.out.println("Language is " + lang.languageCode());
            }

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

- For API details, see DetectDominantLanguage in *AWS SDK for Java 2.x API Reference*.

DetectEntities

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
```

```
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectEntities");
        detectAllEntities(comClient, text);
        comClient.close();
    }

    public static void detectAllEntities(ComprehendClient comClient, String text) {
        try {
            DetectEntitiesRequest detectEntitiesRequest =
 DetectEntitiesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectEntitiesResponse detectEntitiesResult =
 comClient.detectEntities(detectEntitiesRequest);
            List<Entity> entList = detectEntitiesResult.entities();
            for (Entity entity : entList) {
                System.out.println("Entity text is " + entity.text());
            }
```

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

- For API details, see [DetectEntities](#) in *AWS SDK for Java 2.x API Reference*.

DetectKeyPhrases

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
```

```
blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectKeyPhrases");
        detectAllKeyPhrases(comClient, text);
        comClient.close();
    }

    public static void detectAllKeyPhrases(ComprehendClient comClient, String text)
{
        try {
            DetectKeyPhrasesRequest detectKeyPhrasesRequest =
DetectKeyPhrasesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectKeyPhrasesResponse detectKeyPhrasesResult =
comClient.detectKeyPhrases(detectKeyPhrasesRequest);
            List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
            for (KeyPhrase keyPhrase : phraseList) {
                System.out.println("Key phrase text is " + keyPhrase.text());
            }

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

- For API details, see DetectKeyPhrases in *AWS SDK for Java 2.x API Reference*.

DetectSentiment

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/comprehend#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectSentiment {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSentiment");
        detectSentiments(comClient, text);
        comClient.close();
    }

    public static void detectSentiments(ComprehendClient comClient, String text) {
        try {
            DetectSentimentRequest detectSentimentRequest =
 DetectSentimentRequest.builder()
```

```
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSentimentResponse detectSentimentResult =
 comClient.detectSentiment(detectSentimentRequest);
            System.out.println("The Neutral value is " +
 detectSentimentResult.sentimentScore().neutral());

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

- For API details, see DetectSentiment in *AWS SDK for Java 2.x API Reference*.

`DetectSyntax`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSyntax");
        detectAllSyntax(comClient, text);
        comClient.close();
    }

    public static void detectAllSyntax(ComprehendClient comClient, String text) {
        try {
            DetectSyntaxRequest detectSyntaxRequest = DetectSyntaxRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSyntaxResponse detectSyntaxResult =
 comClient.detectSyntax(detectSyntaxRequest);
            List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
            for (SyntaxToken token : syntaxTokens) {
                System.out.println("Language is " + token.text());
                System.out.println("Part of speech is " +
 token.partOfSpeech().tagAsString());
            }

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

- For API details, see DetectSyntax in *AWS SDK for Java 2.x API Reference*.

DynamoDB examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with DynamoDB.

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

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

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

Get started

Hello DynamoDB

The following code examples show how to get started using DynamoDB.

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }
```

```
                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }

            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
                System.exit(1);
            }
        }
        System.out.println("\nDone!");
    }
}
```

- For API details, see ListTables in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios
- Serverless examples

Actions

`BatchGetItem`

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

SDK for Java 2.x

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

shows how to get batch items using the service client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```

```
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchReadItems {
    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItems(dynamoDbClient, tableName);
    }

    public static void getBatchItems(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());
```

```
        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Make the batchGetItem request.
        BatchGetItemResponse batchGetItemResponse =
 dynamoDbClient.batchGetItem(batchGetItemRequest);

        // Extract and print the retrieved items.
        Map<String, List<Map<String, AttributeValue>>> responses =
 batchGetItemResponse.responses();
        if (responses.containsKey(tableName)) {
            List<Map<String, AttributeValue>> musicItems = responses.get(tableName);
            for (Map<String, AttributeValue> item : musicItems) {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            }
        } else {
            System.out.println("No items retrieved.");
        }
    }
}
```

shows how to get batch items using the service client and a paginator.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```
public class BatchGetItemsPaginator {

    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItemsPaginator(dynamoDbClient, tableName) ;
    }

    public static void getBatchItemsPaginator(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Use batchGetItemPaginator for paginated requests.
        dynamoDbClient.batchGetItemPaginator(batchGetItemRequest).stream()
```

```
            .flatMap(response -> response.responses().getOrDefault(tableName,
 Collections.emptyList()).stream())
            .forEach(item -> {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            });
    }
}
```

- For API details, see BatchGetItem in *AWS SDK for Java 2.x API Reference*.

`BatchWriteItem`

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

SDK for Java 2.x

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

Inserts many items into a table by using the service client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutRequest;
import software.amazon.awssdk.services.dynamodb.model.WriteRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchWriteItems {
    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        addBatchItems(dynamoDbClient, tableName);
    }

    public static void addBatchItems(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Specify the updates you want to perform.
        List<WriteRequest> writeRequests = new ArrayList<>();

        // Set item 1.
        Map<String, AttributeValue> item1Attributes = new HashMap<>();
        item1Attributes.put("Artist",
 AttributeValue.builder().s("Artist1").build());
        item1Attributes.put("Rating", AttributeValue.builder().s("5").build());
        item1Attributes.put("Comments", AttributeValue.builder().s("Great
 song!").build());
        item1Attributes.put("SongTitle",
 AttributeValue.builder().s("SongTitle1").build());

 writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item1Attribut

        // Set item 2.
        Map<String, AttributeValue> item2Attributes = new HashMap<>();
```

```
        item2Attributes.put("Artist",
 AttributeValue.builder().s("Artist2").build());
        item2Attributes.put("Rating", AttributeValue.builder().s("4").build());
        item2Attributes.put("Comments", AttributeValue.builder().s("Nice
 melody.").build());
        item2Attributes.put("SongTitle",
 AttributeValue.builder().s("SongTitle2").build());

 writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item2Attribut

        try {
            // Create the BatchWriteItemRequest.
            BatchWriteItemRequest batchWriteItemRequest =
 BatchWriteItemRequest.builder()
                .requestItems(Map.of(tableName, writeRequests))
                .build();

            // Execute the BatchWriteItem operation.
            BatchWriteItemResponse batchWriteItemResponse =
 dynamoDbClient.batchWriteItem(batchWriteItemRequest);

            // Process the response.
            System.out.println("Batch write successful: " + batchWriteItemResponse);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Inserts many items into a table by using the enhanced client.

```
import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/*
 * Before running this code example, create an Amazon DynamoDB table named Customer
 with these columns:
 *   - id - the id of the record that is the key
 *   - custName - the customer name
 *   - email - the email value
 *   - registrationDate - an instant value when the item was added to the table
 *
 * Also, ensure that you have set up your development environment, including your
 credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnhancedBatchWriteItems {
        public static void main(String[] args) {
                Region region = Region.US_EAST_1;
                DynamoDbClient ddb = DynamoDbClient.builder()
                                .region(region)
                                .build();
                DynamoDbEnhancedClient enhancedClient =
 DynamoDbEnhancedClient.builder()
                                .dynamoDbClient(ddb)
                                .build();
                putBatchRecords(enhancedClient);
                ddb.close();
        }

        public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {
                try {
                        DynamoDbTable<Customer> customerMappedTable =
 enhancedClient.table("Customer",
                                        TableSchema.fromBean(Customer.class));
                        DynamoDbTable<Music> musicMappedTable =
 enhancedClient.table("Music",
                                        TableSchema.fromBean(Music.class));
```

```
                        LocalDate localDate = LocalDate.parse("2020-04-07");
                        LocalDateTime localDateTime = localDate.atStartOfDay();
                        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

                        Customer record2 = new Customer();
                        record2.setCustName("Fred Pink");
                        record2.setId("id110");
                        record2.setEmail("fredp@noserver.com");
                        record2.setRegistrationDate(instant);

                        Customer record3 = new Customer();
                        record3.setCustName("Susan Pink");
                        record3.setId("id120");
                        record3.setEmail("spink@noserver.com");
                        record3.setRegistrationDate(instant);

                        Customer record4 = new Customer();
                        record4.setCustName("Jerry orange");
                        record4.setId("id101");
                        record4.setEmail("jorange@noserver.com");
                        record4.setRegistrationDate(instant);

                        BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest
 = BatchWriteItemEnhancedRequest
                                        .builder()
                                        .writeBatches(

 WriteBatch.builder(Customer.class) // add items to the Customer

      // table

 .mappedTableResource(customerMappedTable)

 .addPutItem(builder -> builder.item(record2))

 .addPutItem(builder -> builder.item(record3))

 .addPutItem(builder -> builder.item(record4))
                                                                      .build(),

 WriteBatch.builder(Music.class) // delete an item from the Music

   // table
```

```
.mappedTableResource(musicMappedTable)

.addDeleteItem(builder -> builder.key(

   Key.builder().partitionValue(

                "Famous Band")

                .build()))
                                                    .build())
                                 .build();

                    // Add three items to the Customer table and delete one item
from the Music
                    // table.

enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);
                    System.out.println("done");

            } catch (DynamoDbException e) {
                    System.err.println(e.getMessage());
                    System.exit(1);
            }
        }
}
```

- For API details, see BatchWriteItem in *AWS SDK for Java 2.x API Reference*.

CreateTable

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key>

                Where:
                    tableName - The Amazon DynamoDB table to create (for example,
 Music3).
                    key - The key for the Amazon DynamoDB table (for example,
 Artist).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
```

```
        String key = args[1];
        System.out.println("Creating an Amazon DynamoDB table " + tableName + " with
 a simple primary key: " + key);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        String result = createTable(ddb, tableName, key);
        System.out.println("New table is " + result);
        ddb.close();
    }

    public static String createTable(DynamoDbClient ddb, String tableName, String
 key) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        CreateTableRequest request = CreateTableRequest.builder()
                .attributeDefinitions(AttributeDefinition.builder()
                        .attributeName(key)
                        .attributeType(ScalarAttributeType.S)
                        .build())
                .keySchema(KeySchemaElement.builder()
                        .attributeName(key)
                        .keyType(KeyType.HASH)
                        .build())
                .provisionedThroughput(ProvisionedThroughput.builder()
                        .readCapacityUnits(10L)
                        .writeCapacityUnits(10L)
                        .build())
                .tableName(tableName)
                .build();

        String newTable;
        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            newTable = response.tableDescription().tableName();
```

```
            return newTable;

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see CreateTable in *AWS SDK for Java 2.x API Reference*.

DeleteItem

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteItem {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                    <tableName> <key> <keyval>

                Where:
                    tableName - The Amazon DynamoDB table to delete the item from
 (for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).\s
                    keyval - The key value that represents the item to delete (for
 example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Deleting item \"%s\" from %s\n", keyVal, tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        deleteDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName,
 String key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        DeleteItemRequest deleteReq = DeleteItemRequest.builder()
                .tableName(tableName)
                .key(keyToGet)
                .build();
```

```
        try {
            ddb.deleteItem(deleteReq);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteItem in *AWS SDK for Java 2.x API Reference*.

DeleteTable

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteTable {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to delete (for example,
 Music3).

                **Warning** This program will delete the table that you specify!
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Deleting the Amazon DynamoDB table %s...\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }
}
```

- For API details, see DeleteTable in *AWS SDK for Java 2.x API Reference*.

DescribeTable

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

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/dynamodb#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import
 software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to get information about
 (for example, Music3).
                """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Getting description for %s\n\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        describeDymamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void describeDymamoDBTable(DynamoDbClient ddb, String tableName) {
        DescribeTableRequest request = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            TableDescription tableInfo = ddb.describeTable(request).table();
            if (tableInfo != null) {
                System.out.format("Table name  : %s\n", tableInfo.tableName());
                System.out.format("Table ARN   : %s\n", tableInfo.tableArn());
                System.out.format("Status      : %s\n", tableInfo.tableStatus());
                System.out.format("Item count  : %d\n", tableInfo.itemCount());
                System.out.format("Size (bytes): %d\n", tableInfo.tableSizeBytes());

                ProvisionedThroughputDescription throughputInfo =
 tableInfo.provisionedThroughput();
                System.out.println("Throughput");
                System.out.format("  Read Capacity : %d\n",
 throughputInfo.readCapacityUnits());
                System.out.format("  Write Capacity: %d\n",
 throughputInfo.writeCapacityUnits());

                List<AttributeDefinition> attributes =
 tableInfo.attributeDefinitions();
                System.out.println("Attributes");
                for (AttributeDefinition a : attributes) {
                    System.out.format("  %s (%s)\n", a.attributeName(),
 a.attributeType());
                }
```

```
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("\nDone!");
    }
}
```

- For API details, see DescribeTable in *AWS SDK for Java 2.x API Reference*.

DescribeTimeToLive

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

SDK for Java 2.x

Describe TTL configuration on an existing DynamoDB table.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.Optional;

        final DescribeTimeToLiveRequest request =
 DescribeTimeToLiveRequest.builder()
                .tableName(tableName)
                .build();
        try (DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build()) {
            final DescribeTimeToLiveResponse response =
 ddb.describeTimeToLive(request);
            System.out.println(tableName + " description of time to live is "
                    + response.toString());
        } catch (ResourceNotFoundException e) {
```

```
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.exit(0);
```

- For API details, see DescribeTimeToLive in *AWS SDK for Java 2.x API Reference*.

`GetItem`

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

SDK for Java 2.x

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

Gets an item from a table by using the DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 *
 * To get an item from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, see the EnhancedGetItem example.
 */
public class GetItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal>

                Where:
                    tableName - The Amazon DynamoDB table from which an item is
 retrieved (for example, Music3).\s
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).\s
                    keyval - The key value that represents the item to get (for
 example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Retrieving item \"%s\" from \"%s\"\n", keyVal,
 tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        getDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
 key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
```

```
                .s(keyVal)
                .build());

        GetItemRequest request = GetItemRequest.builder()
                .key(keyToGet)
                .tableName(tableName)
                .build();

        try {
            // If there is no matching item, GetItem does not return any data.
            Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();
            if (returnedItem.isEmpty())
                System.out.format("No item found with the key %s!\n", key);
            else {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");
                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see GetItem in *AWS SDK for Java 2.x API Reference*.

`ListTables`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
```

```
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }

            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
                System.exit(1);
            }
        }
        System.out.println("\nDone!");
    }
}
```

- For API details, see ListTables in *AWS SDK for Java 2.x API Reference*.

PutItem

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

SDK for Java 2.x

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

Puts an item into a table using DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```

```
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To place items into an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedPutItem example.
 */
public class PutItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal> <albumtitle> <albumtitleval> <awards>
 <awardsval> <Songtitle> <songtitleval>

                Where:
                    tableName - The Amazon DynamoDB table in which an item is placed
 (for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).
                    keyval - The key value that represents the item to get (for
 example, Famous Band).
                    albumTitle - The Album title (for example, AlbumTitle).
                    AlbumTitleValue - The name of the album (for example, Songs
 About Life ).
                    Awards - The awards column (for example, Awards).
                    AwardVal - The value of the awards (for example, 10).
                    SongTitle - The song title (for example, SongTitle).
                    SongTitleVal - The value of the song title (for example, Happy
 Day).
                **Warning** This program will  place an item that you specify into a
 table!
```

```
            """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String albumTitle = args[3];
        String albumTitleValue = args[4];
        String awards = args[5];
        String awardVal = args[6];
        String songTitle = args[7];
        String songTitleVal = args[8];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        putItemInTable(ddb, tableName, key, keyVal, albumTitle, albumTitleValue,
 awards, awardVal, songTitle,
                songTitleVal);
        System.out.println("Done!");
        ddb.close();
    }

    public static void putItemInTable(DynamoDbClient ddb,
            String tableName,
            String key,
            String keyVal,
            String albumTitle,
            String albumTitleValue,
            String awards,
            String awardVal,
            String songTitle,
            String songTitleVal) {

        HashMap<String, AttributeValue> itemValues = new HashMap<>();
        itemValues.put(key, AttributeValue.builder().s(keyVal).build());
        itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
```

```
        itemValues.put(albumTitle,
 AttributeValue.builder().s(albumTitleValue).build());
        itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

        PutItemRequest request = PutItemRequest.builder()
                .tableName(tableName)
                .item(itemValues)
                .build();

        try {
            PutItemResponse response = ddb.putItem(request);
            System.out.println(tableName + " was successfully updated. The request
 id is "
                    + response.responseMetadata().requestId());

        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            System.err.println("Be sure that it exists and that you've typed its
 name correctly!");
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see PutItem in *AWS SDK for Java 2.x API Reference*.

`Query`

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

SDK for Java 2.x

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

Queries a table by using [DynamoDbClient](https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/dynamodb/DynamoDbClient.html).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To query items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedQueryRecords example.
 */
public class Query {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <partitionKeyName> <partitionKeyVal>

                Where:
                    tableName - The Amazon DynamoDB table to put the item in (for
 example, Music3).
                    partitionKeyName - The partition key name of the Amazon DynamoDB
 table (for example, Artist).
                    partitionKeyVal - The value of the partition key that should
 match (for example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
```

```
        String partitionKeyName = args[1];
        String partitionKeyVal = args[2];

        // For more information about an alias, see:
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.ExpressionAttributeNames.html
        String partitionAlias = "#a";

        System.out.format("Querying %s", tableName);
        System.out.println("");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        int count = queryTable(ddb, tableName, partitionKeyName, partitionKeyVal,
 partitionAlias);
        System.out.println("There were " + count + "  record(s) returned");
        ddb.close();
    }

    public static int queryTable(DynamoDbClient ddb, String tableName, String
 partitionKeyName, String partitionKeyVal,
            String partitionAlias) {
        // Set up an alias for the partition key name in case it's a reserved word.
        HashMap<String, String> attrNameAlias = new HashMap<String, String>();
        attrNameAlias.put(partitionAlias, partitionKeyName);

        // Set up mapping of the partition name with the value.
        HashMap<String, AttributeValue> attrValues = new HashMap<>();
        attrValues.put(":" + partitionKeyName, AttributeValue.builder()
                .s(partitionKeyVal)
                .build());

        QueryRequest queryReq = QueryRequest.builder()
                .tableName(tableName)
                .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
                .expressionAttributeNames(attrNameAlias)
                .expressionAttributeValues(attrValues)
                .build();

        try {
            QueryResponse response = ddb.query(queryReq);
            return response.count();
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return -1;
    }
}
```

Queries a table by using `DynamoDbClient` and a secondary index.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Create the Movies table by running the Scenario example and loading the Movie
 * data from the JSON file. Next create a secondary
 * index for the Movies table that uses only the year column. Name the index
 * **year-index**. For more information, see:
 *
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html
 */
public class QueryItemsUsingIndex {
    public static void main(String[] args) {
        String tableName = "Movies";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
```

```
        queryIndex(ddb, tableName);
        ddb.close();
    }

    public static void queryIndex(DynamoDbClient ddb, String tableName) {
        try {
            Map<String, String> expressionAttributesNames = new HashMap<>();
            expressionAttributesNames.put("#year", "year");
            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
            expressionAttributeValues.put(":yearValue",
 AttributeValue.builder().n("2013").build());

            QueryRequest request = QueryRequest.builder()
                    .tableName(tableName)
                    .indexName("year-index")
                    .keyConditionExpression("#year = :yearValue")
                    .expressionAttributeNames(expressionAttributesNames)
                    .expressionAttributeValues(expressionAttributeValues)
                    .build();

            System.out.println("=== Movie Titles ===");
            QueryResponse response = ddb.query(request);
            response.items()
                    .forEach(movie -> System.out.println(movie.get("title").s()));

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see Query in *AWS SDK for Java 2.x API Reference*.

Scan

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

SDK for Java 2.x

Note

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

Scans an Amazon DynamoDB table using DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To scan items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, See the EnhancedScanRecords example.
 */

public class DynamoDBScanItems {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to get information from
 (for example, Music3).
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        scanItems(ddb, tableName);
        ddb.close();
    }

    public static void scanItems(DynamoDbClient ddb, String tableName) {
        try {
            ScanRequest scanRequest = ScanRequest.builder()
                    .tableName(tableName)
                    .build();

            ScanResponse response = ddb.scan(scanRequest);
            for (Map<String, AttributeValue> item : response.items()) {
                Set<String> keys = item.keySet();
                for (String key : keys) {
                    System.out.println("The key name is " + key + "\n");
                    System.out.println("The value is " + item.get(key).s());
                }
            }

        } catch (DynamoDbException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- For API details, see Scan in *AWS SDK for Java 2.x API Reference*.

UpdateItem

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

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/dynamodb#code-examples).

Updates an item in a table using [DynamoDbClient](https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/dynamodb/DynamoDbClient.html).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To update an Amazon DynamoDB table using the AWS SDK for Java V2, its better
 * practice to use the
 * Enhanced Client, See the EnhancedModifyItem example.
 */
public class UpdateItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal> <name> <updateVal>

                Where:
```

```
                    tableName - The Amazon DynamoDB table (for example, Music3).
                    key - The name of the key in the table (for example, Artist).
                    keyVal - The value of the key (for example, Famous Band).
                    name - The name of the column where the value is updated (for
example, Awards).
                    updateVal - The value used to update an item (for example, 14).
                 Example:
                    UpdateItem Music3 Artist Famous Band Awards 14
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String name = args[3];
        String updateVal = args[4];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        updateTableItem(ddb, tableName, key, keyVal, name, updateVal);
        ddb.close();
    }

    public static void updateTableItem(DynamoDbClient ddb,
            String tableName,
            String key,
            String keyVal,
            String name,
            String updateVal) {

        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put(name, AttributeValueUpdate.builder()
                .value(AttributeValue.builder().s(updateVal).build())
```

```
                .action(AttributeAction.PUT)
                .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(itemKey)
                .attributeUpdates(updatedValues)
                .build();

        try {
            ddb.updateItem(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("The Amazon DynamoDB table was updated!");
    }
}
```

- For API details, see UpdateItem in *AWS SDK for Java 2.x API Reference*.

`UpdateTimeToLive`

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

SDK for Java 2.x

Enable TTL on an existing DynamoDB table.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.Optional;

        final TimeToLiveSpecification ttlSpecification =
 TimeToLiveSpecification.builder()
                .attributeName(ttlAttributeName)
```

```
                .enabled(true)
                .build();
        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
                .tableName(tableName)
                .timeToLiveSpecification(ttlSpecification)
                .build();
        try (DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build()) {
                final UpdateTimeToLiveResponse response =
 ddb.updateTimeToLive(request);
                System.out.println(tableName + " had its TTL successfully updated.
 The request id is "
                        + response.responseMetadata().requestId());
        } catch (ResourceNotFoundException e) {
                System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
                System.exit(1);
        } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
                System.exit(1);
        }
        System.out.println("Done!");
```

Disable TTL on an existing DynamoDB table.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.Optional;

        final Region region = Optional.ofNullable(args[2]).isEmpty() ?
 Region.US_EAST_1 : Region.of(args[2]);
        final TimeToLiveSpecification ttlSpecification =
 TimeToLiveSpecification.builder()
                .attributeName(ttlAttributeName)
                .enabled(false)
```

```
                .build();
        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
                .tableName(tableName)
                .timeToLiveSpecification(ttlSpecification)
                .build();
        try (DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build()) {
            final UpdateTimeToLiveResponse response = ddb.updateTimeToLive(request);
            System.out.println(tableName + " had its TTL successfully updated. The
 request id is "
                    + response.responseMetadata().requestId());
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Done!");
```

- For API details, see UpdateTimeToLive in *AWS SDK for Java 2.x API Reference*.

Scenarios

Conditionally update an item's TTL

The following code example shows how to conditionally update an item's TTL.

SDK for Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.amazon.samplelib.ttl;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

public class UpdateTTLConditional {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <tableName> <primaryKey> <sortKey> <newTtlAttribute> <region>
                Where:
                    tableName - The Amazon DynamoDB table being queried.
                    primaryKey - The name of the primary key. Also known as the hash
 or partition key.
                    sortKey - The name of the sort key. Also known as the range
 attribute.
                    newTtlAttribute - New attribute name (as part of the update
 command)
                    region (optional) - The AWS region that the Amazon DynamoDB
 table is located in. (Default: us-east-1)
                """;
        // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
        if (!(args.length == 4 || args.length == 5)) {
            System.out.println(usage);
            System.exit(1);
        }
        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final String newTtlAttribute = args[3];
        Region region = Optional.ofNullable(args[4]).isEmpty() ? Region.US_EAST_1 :
 Region.of(args[4]);

        // Get current time in epoch second format
        final long currentTime = System.currentTimeMillis() / 1000;
        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (90 * 24 * 60 * 60);
        // An expression that defines one or more attributes to be updated, the
 action to be performed on them, and new values for them.
        final String updateExpression = "SET newTtlAttribute = :val1";
        // A condition that must be satisfied in order for a conditional update to
 succeed.
```

```
        final String conditionExpression = "expireAt > :val2";

        final ImmutableMap<String, AttributeValue> keyMap =
                ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
                        "sortKey", AttributeValue.fromS(sortKey));
        final Map<String, AttributeValue> expressionAttributeValues =
 ImmutableMap.of(
                ":val1", AttributeValue.builder().s(newTtlAttribute).build(),
                ":val2",
 AttributeValue.builder().s(String.valueOf(expireDate)).build()
        );

        final UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(keyMap)
                .updateExpression(updateExpression)
                .conditionExpression(conditionExpression)
                .expressionAttributeValues(expressionAttributeValues)
                .build();
        try (DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build()) {
            final UpdateItemResponse response = ddb.updateItem(request);
            System.out.println(tableName + " UpdateItem operation with conditional
 TTL successful. Request id is "
                    + response.responseMetadata().requestId());
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.exit(0);
    }
}
```

- For API details, see UpdateItem in *AWS SDK for Java 2.x API Reference*.

Create an item with a TTL

The following code example shows how to create an item with TTL.

SDK for Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

package com.amazon.samplelib.ttl;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.utils.ImmutableMap;

import java.io.Serializable;
import java.util.Map;
import java.util.Optional;

public class CreateTTL {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <tableName> <primaryKey> <sortKey> <region>
                Where:
                    tableName - The Amazon DynamoDB table being queried.
                    primaryKey - The name of the primary key. Also known as the hash
 or partition key.
                    sortKey - The name of the sort key. Also known as the range
 attribute.
                    region (optional) - The AWS region that the Amazon DynamoDB
 table is located in. (Default: us-east-1)
                """;
        // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
        if (!(args.length == 3 || args.length == 4)) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String tableName = args[0];
        String primaryKey = args[1];
        String sortKey = args[2];
        Region region = Optional.ofNullable(args[3]).isEmpty() ? Region.US_EAST_1 :
 Region.of(args[3]);

        // Get current time in epoch second format
        final long createDate = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = createDate + (90 * 24 * 60 * 60);

        final ImmutableMap<String, ? extends Serializable> itemMap =
                ImmutableMap.of("primaryKey", primaryKey,
                "sortKey", sortKey,
                "creationDate", createDate,
                "expireAt", expireDate);
        final PutItemRequest request = PutItemRequest.builder()
                .tableName(tableName)
                .item((Map<String, AttributeValue>) itemMap)
                .build();
        try (DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build()) {
            final PutItemResponse response = ddb.putItem(request);
            System.out.println(tableName + " PutItem operation with TTL successful.
 Request id is "
                    + response.responseMetadata().requestId());
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.exit(0);
    }
}
```

- For API details, see PutItem in *AWS SDK for Java 2.x API Reference*.

Get started with 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, then delete the table.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/dynamodb#readme).

Create a DynamoDB table.

```
    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName("year")
                .attributeType("N")
                .build());

        attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName("title")
                .attributeType("S")
                .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
                .attributeName("year")
```

```
                .keyType(KeyType.HASH)
                .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
                .attributeName("title")
                .keyType(KeyType.RANGE)
                .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
                .keySchema(tableKey)
                .provisionedThroughput(ProvisionedThroughput.builder()
                        .readCapacityUnits(10L)
                        .writeCapacityUnits(10L)
                        .build())
                .attributeDefinitions(attributeDefinitions)
                .tableName(tableName)
                .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Create a helper function to download and extract the sample JSON file.

```
    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String
 fileName) throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
 TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }
```

Get an item from a table.

```
    public static void getItem(DynamoDbClient ddb) {

        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
```

```
                .n("1933")
                .build());

        keyToGet.put("title", AttributeValue.builder()
                .s("King Kong")
                .build());

        GetItemRequest request = GetItemRequest.builder()
                .key(keyToGet)
                .tableName("Movies")
                .build();

        try {
            Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Full example.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 *
 * This Java example performs these tasks:
 *
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the
 * Enhanced client.
 * 3. Gets data from the Movie table.
 * 4. Adds a new item.
 * 5. Updates an item.
 * 6. Uses a Scan to query items using the Enhanced client.
 * 7. Queries all items where the year is 2013 using the Enhanced Client.
 * 8. Deletes the table.
 */

public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <fileName>

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

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = "Movies";
        String fileName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println(
                "1. Creating an Amazon DynamoDB table named Movies with a key named
 year and a sort key named title.");
        createTable(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Loading data into the Amazon DynamoDB table.");
        loadData(ddb, tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Getting data from the Movie table.");
        getItem(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Putting a record into the Amazon DynamoDB table.");
        putRecord(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Updating a record.");
        updateTableItem(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Scanning the Amazon DynamoDB table.");
        scanMovies(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Querying the Movies released in 2013.");
        queryTable(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }
```

```
    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName("year")
                .attributeType("N")
                .build());

        attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName("title")
                .attributeType("S")
                .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
                .attributeName("year")
                .keyType(KeyType.HASH)
                .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
                .attributeName("title")
                .keyType(KeyType.RANGE)
                .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
                .keySchema(tableKey)
                .provisionedThroughput(ProvisionedThroughput.builder()
                        .readCapacityUnits(10L)
                        .writeCapacityUnits(10L)
                        .build())
                .attributeDefinitions(attributeDefinitions)
                .tableName(tableName)
                .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
```

```
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Query the table.
    public static void queryTable(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                    .dynamoDbClient(ddb)
                    .build();

            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
 TableSchema.fromBean(Movies.class));
            QueryConditional queryConditional = QueryConditional
                    .keyEqualTo(Key.builder()
                            .partitionValue(2013)
                            .build());

            // Get items in the table and write out the ID value.
            Iterator<Movies> results =
 custTable.query(queryConditional).items().iterator();
            String result = "";

            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The title of the movie is " + rec.getTitle());
                System.out.println("The movie information  is " + rec.getInfo());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
```

```
            System.exit(1);
        }
    }

    // Scan the table.
    public static void scanMovies(DynamoDbClient ddb, String tableName) {
        System.out.println("******* Scanning all movies.\n");
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                    .dynamoDbClient(ddb)
                    .build();

            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            Iterator<Movies> results = custTable.scan().items().iterator();
            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The movie title is " + rec.getTitle());
                System.out.println("The movie year is " + rec.getYear());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
```

```
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }

    // Update the record to include show only directors.
    public static void updateTableItem(DynamoDbClient ddb, String tableName) {
        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put("year", AttributeValue.builder().n("1933").build());
        itemKey.put("title", AttributeValue.builder().s("King Kong").build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put("info", AttributeValueUpdate.builder()
                .value(AttributeValue.builder().s("{\"directors\":[\"Merian C.
 Cooper\",\"Ernest B. Schoedsack\"]")
                        .build())
                .action(AttributeAction.PUT)
                .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(itemKey)
                .attributeUpdates(updatedValues)
                .build();

        try {
            ddb.updateItem(request);
        } catch (ResourceNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Item was updated!");
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    public static void putRecord(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                    .dynamoDbClient(ddb)
                    .build();

            DynamoDbTable<Movies> table = enhancedClient.table("Movies",
 TableSchema.fromBean(Movies.class));

            // Populate the Table.
            Movies record = new Movies();
            record.setYear(2020);
            record.setTitle("My Movie2");
            record.setInfo("no info");
            table.putItem(record);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Added a new movie to the table.");
```

```
    }

    public static void getItem(DynamoDbClient ddb) {

        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
                .n("1933")
                .build());

        keyToGet.put("title", AttributeValue.builder()
                .s("King Kong")
                .build());

        GetItemRequest request = GetItemRequest.builder()
                .key(keyToGet)
                .tableName("Movies")
                .build();

        try {
            Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x 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 Java 2.x

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

```
public class ScenarioPartiQLBatch {
        public static void main(String[] args) throws IOException {
                String tableName = "MoviesPartiQBatch";
                Region region = Region.US_EAST_1;
                DynamoDbClient ddb = DynamoDbClient.builder()
                                .region(region)
                                .build();
```

```
                System.out.println("******* Creating an Amazon DynamoDB table named
 " + tableName
                                + " with a key named year and a sort key named
 title.");
                createTable(ddb, tableName);

                System.out.println("******* Adding multiple records into the " +
 tableName
                                + " table using a batch command.");
                putRecordBatch(ddb);

                System.out.println("******* Updating multiple records using a batch
 command.");
                updateTableItemBatch(ddb);

                System.out.println("******* Deleting multiple records using a batch
 command.");
                deleteItemBatch(ddb);

                System.out.println("******* Deleting the Amazon DynamoDB table.");
                deleteDynamoDBTable(ddb, tableName);
                ddb.close();
        }

        public static void createTable(DynamoDbClient ddb, String tableName) {
                DynamoDbWaiter dbWaiter = ddb.waiter();
                ArrayList<AttributeDefinition> attributeDefinitions = new
 ArrayList<>();

                // Define attributes.
                attributeDefinitions.add(AttributeDefinition.builder()
                                .attributeName("year")
                                .attributeType("N")
                                .build());

                attributeDefinitions.add(AttributeDefinition.builder()
                                .attributeName("title")
                                .attributeType("S")
                                .build());

                ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
                KeySchemaElement key = KeySchemaElement.builder()
                                .attributeName("year")
                                .keyType(KeyType.HASH)
```

```
                                .build();

                KeySchemaElement key2 = KeySchemaElement.builder()
                                .attributeName("title")
                                .keyType(KeyType.RANGE) // Sort
                                .build();

                // Add KeySchemaElement objects to the list.
                tableKey.add(key);
                tableKey.add(key2);

                CreateTableRequest request = CreateTableRequest.builder()
                                .keySchema(tableKey)

.provisionedThroughput(ProvisionedThroughput.builder()
                                                .readCapacityUnits(new Long(10))
                                                .writeCapacityUnits(new Long(10))
                                                .build())
                                .attributeDefinitions(attributeDefinitions)
                                .tableName(tableName)
                                .build();

                try {
                        CreateTableResponse response = ddb.createTable(request);
                        DescribeTableRequest tableRequest =
DescribeTableRequest.builder()
                                        .tableName(tableName)
                                        .build();

                        // Wait until the Amazon DynamoDB table is created.
                        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter
                                        .waitUntilTableExists(tableRequest);

waiterResponse.matched().response().ifPresent(System.out::println);
                        String newTable = response.tableDescription().tableName();
                        System.out.println("The " + newTable + " was successfully
created.");

                } catch (DynamoDbException e) {
                        System.err.println(e.getMessage());
                        System.exit(1);
                }
        }
```

```
    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE
{'year':?, 'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                    .n(String.valueOf("2022"))
                    .build();

            AttributeValue att2 = AttributeValue.builder()
                    .s("My Movie 1")
                    .build();

            AttributeValue att3 = AttributeValue.builder()
                    .s("No Information")
                    .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            BatchStatementRequest statementRequestMovie1 =
BatchStatementRequest.builder()
                    .statement(sqlStatement)
                    .parameters(parameters)
                    .build();

            // Set data for Movie 2.
            List<AttributeValue> parametersMovie2 = new ArrayList<>();
            AttributeValue attMovie2 = AttributeValue.builder()
                    .n(String.valueOf("2022"))
                    .build();

            AttributeValue attMovie2A = AttributeValue.builder()
                    .s("My Movie 2")
                    .build();

            AttributeValue attMovie2B = AttributeValue.builder()
                    .s("No Information")
                    .build();
```

```
                        parametersMovie2.add(attMovie2);
                        parametersMovie2.add(attMovie2A);
                        parametersMovie2.add(attMovie2B);

                        BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
                                        .statement(sqlStatement)
                                        .parameters(parametersMovie2)
                                        .build();

                        // Set data for Movie 3.
                        List<AttributeValue> parametersMovie3 = new ArrayList<>();
                        AttributeValue attMovie3 = AttributeValue.builder()
                                        .n(String.valueOf("2022"))
                                        .build();

                        AttributeValue attMovie3A = AttributeValue.builder()
                                        .s("My Movie 3")
                                        .build();

                        AttributeValue attMovie3B = AttributeValue.builder()
                                        .s("No Information")
                                        .build();

                        parametersMovie3.add(attMovie3);
                        parametersMovie3.add(attMovie3A);
                        parametersMovie3.add(attMovie3B);

                        BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
                                        .statement(sqlStatement)
                                        .parameters(parametersMovie3)
                                        .build();

                        // Add all three movies to the list.
                        List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
                        myBatchStatementList.add(statementRequestMovie1);
                        myBatchStatementList.add(statementRequestMovie2);
                        myBatchStatementList.add(statementRequestMovie3);

                        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
```

```
                                        .statements(myBatchStatementList)
                                        .build();

                        BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
                        System.out.println("ExecuteStatement successful: " +
response.toString());
                        System.out.println("Added new movies using a batch
command.");

                } catch (DynamoDbException e) {
                        System.err.println(e.getMessage());
                        System.exit(1);
                }
        }

        public static void updateTableItemBatch(DynamoDbClient ddb) {
                String sqlStatement = "UPDATE MoviesPartiQBatch SET info =
'directors\":[\"Merian C. Cooper\",\"Ernest B. Schoedsack' where year=? and
title=?";
                List<AttributeValue> parametersRec1 = new ArrayList<>();

                // Update three records.
                AttributeValue att1 = AttributeValue.builder()
                                .n(String.valueOf("2022"))
                                .build();

                AttributeValue att2 = AttributeValue.builder()
                                .s("My Movie 1")
                                .build();

                parametersRec1.add(att1);
                parametersRec1.add(att2);

                BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
                                .statement(sqlStatement)
                                .parameters(parametersRec1)
                                .build();

                // Update record 2.
                List<AttributeValue> parametersRec2 = new ArrayList<>();
                AttributeValue attRec2 = AttributeValue.builder()
                                .n(String.valueOf("2022"))
```

```
                                .build();

                AttributeValue attRec2a = AttributeValue.builder()
                                .s("My Movie 2")
                                .build();

                parametersRec2.add(attRec2);
                parametersRec2.add(attRec2a);
                BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
                                .statement(sqlStatement)
                                .parameters(parametersRec2)
                                .build();

                // Update record 3.
                List<AttributeValue> parametersRec3 = new ArrayList<>();
                AttributeValue attRec3 = AttributeValue.builder()
                                .n(String.valueOf("2022"))
                                .build();

                AttributeValue attRec3a = AttributeValue.builder()
                                .s("My Movie 3")
                                .build();

                parametersRec3.add(attRec3);
                parametersRec3.add(attRec3a);
                BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
                                .statement(sqlStatement)
                                .parameters(parametersRec3)
                                .build();

                // Add all three movies to the list.
                List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
                myBatchStatementList.add(statementRequestRec1);
                myBatchStatementList.add(statementRequestRec2);
                myBatchStatementList.add(statementRequestRec3);

                BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
                                .statements(myBatchStatementList)
                                .build();
```

```
                try {
                        BatchExecuteStatementResponse response =
 ddb.batchExecuteStatement(batchRequest);
                        System.out.println("ExecuteStatement successful: " +
 response.toString());
                        System.out.println("Updated three movies using a batch
 command.");

                } catch (DynamoDbException e) {
                        System.err.println(e.getMessage());
                        System.exit(1);
                }
                System.out.println("Item was updated!");
        }

        public static void deleteItemBatch(DynamoDbClient ddb) {
                String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ?
 and title=?";
                List<AttributeValue> parametersRec1 = new ArrayList<>();

                // Specify three records to delete.
                AttributeValue att1 = AttributeValue.builder()
                                .n(String.valueOf("2022"))
                                .build();

                AttributeValue att2 = AttributeValue.builder()
                                .s("My Movie 1")
                                .build();

                parametersRec1.add(att1);
                parametersRec1.add(att2);

                BatchStatementRequest statementRequestRec1 =
 BatchStatementRequest.builder()
                                .statement(sqlStatement)
                                .parameters(parametersRec1)
                                .build();

                // Specify record 2.
                List<AttributeValue> parametersRec2 = new ArrayList<>();
                AttributeValue attRec2 = AttributeValue.builder()
                                .n(String.valueOf("2022"))
                                .build();
```

```
                AttributeValue attRec2a = AttributeValue.builder()
                                .s("My Movie 2")
                                .build();

                parametersRec2.add(attRec2);
                parametersRec2.add(attRec2a);
                BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
                                .statement(sqlStatement)
                                .parameters(parametersRec2)
                                .build();

                // Specify record 3.
                List<AttributeValue> parametersRec3 = new ArrayList<>();
                AttributeValue attRec3 = AttributeValue.builder()
                                .n(String.valueOf("2022"))
                                .build();

                AttributeValue attRec3a = AttributeValue.builder()
                                .s("My Movie 3")
                                .build();

                parametersRec3.add(attRec3);
                parametersRec3.add(attRec3a);

                BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
                                .statement(sqlStatement)
                                .parameters(parametersRec3)
                                .build();

                // Add all three movies to the list.
                List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
                myBatchStatementList.add(statementRequestRec1);
                myBatchStatementList.add(statementRequestRec2);
                myBatchStatementList.add(statementRequestRec3);

                BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
                                .statements(myBatchStatementList)
                                .build();

                try {
```

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

                } catch (DynamoDbException e) {
                        System.err.println(e.getMessage());
                        System.exit(1);
                }
        }

        public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
 {
                DeleteTableRequest request = DeleteTableRequest.builder()
                                .tableName(tableName)
                                .build();

                try {
                        ddb.deleteTable(request);

                } catch (DynamoDbException e) {
                        System.err.println(e.getMessage());
                        System.exit(1);
                }
                System.out.println(tableName + " was successfully deleted!");
        }

        private static ExecuteStatementResponse
 executeStatementRequest(DynamoDbClient ddb, String statement,
                        List<AttributeValue> parameters) {
                ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                                .statement(statement)
                                .parameters(parameters)
                                .build();

                return ddb.executeStatement(request);
        }
}
```

- For API details, see BatchExecuteStatement in *AWS SDK for Java 2.x 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 Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/dynamodb#code-examples).

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <fileName>

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

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String fileName = args[0];
        String tableName = "MoviesPartiQ";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
```

```
        System.out.println(
                "******* Creating an Amazon DynamoDB table named MoviesPartiQ with a
 key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("******* Loading data into the MoviesPartiQ table.");
        loadData(ddb, fileName);

        System.out.println("******* Getting data from the MoviesPartiQ table.");
        getItem(ddb);

        System.out.println("******* Putting a record into the MoviesPartiQ table.");
        putRecord(ddb);

        System.out.println("******* Updating a record.");
        updateTableItem(ddb);

        System.out.println("******* Querying the movies released in 2013.");
        queryTable(ddb);

        System.out.println("******* Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName("year")
                .attributeType("N")
                .build());

        attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName("title")
                .attributeType("S")
                .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
                .attributeName("year")
```

```
                .keyType(KeyType.HASH)
                .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
                .attributeName("title")
                .keyType(KeyType.RANGE) // Sort
                .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
                .keySchema(tableKey)
                .provisionedThroughput(ProvisionedThroughput.builder()
                        .readCapacityUnits(new Long(10))
                        .writeCapacityUnits(new Long(10))
                        .build())
                .attributeDefinitions(attributeDefinitions)
                .tableName(tableName)
                .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String fileName) throws
 IOException {
```

```
        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        List<AttributeValue> parameters = new ArrayList<>();
        while (iter.hasNext()) {

            // Add 200 movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            AttributeValue att1 = AttributeValue.builder()
                    .n(String.valueOf(year))
                    .build();

            AttributeValue att2 = AttributeValue.builder()
                    .s(title)
                    .build();

            AttributeValue att3 = AttributeValue.builder()
                    .s(info)
                    .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            // Insert the movie into the Amazon DynamoDB table.
            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added Movie " + title);

            parameters.remove(att1);
            parameters.remove(att2);
            parameters.remove(att3);
```

```
            t++;
        }
    }

    public static void getItem(DynamoDbClient ddb) {

        String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
                .n("2012")
                .build();

        AttributeValue att2 = AttributeValue.builder()
                .s("The Perks of Being a Wallflower")
                .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            ExecuteStatementResponse response = executeStatementRequest(ddb,
 sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
 response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecord(DynamoDbClient ddb) {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
 'title' : ?, 'info' : ?}";
        try {
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                    .n(String.valueOf("2020"))
                    .build();

            AttributeValue att2 = AttributeValue.builder()
                    .s("My Movie")
```

```
                .build();

            AttributeValue att3 = AttributeValue.builder()
                    .s("No Information")
                    .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added new movie.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItem(DynamoDbClient ddb) {

        String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian
 C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2013"))
                .build();

        AttributeValue att2 = AttributeValue.builder()
                .s("The East")
                .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            executeStatementRequest(ddb, sqlStatement, parameters);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }
```

```
    // Query the table where the year is 2013.
    public static void queryTable(DynamoDbClient ddb) {
        String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
 year";
        try {

            List<AttributeValue> parameters = new ArrayList<>();
            AttributeValue att1 = AttributeValue.builder()
                    .n(String.valueOf("2013"))
                    .build();
            parameters.add(att1);

            // Get items in the table and write out the ID value.
            ExecuteStatementResponse response = executeStatementRequest(ddb,
 sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
 response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

        DeleteTableRequest request = DeleteTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
 ddb, String statement,
            List<AttributeValue> parameters) {
```

```
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .statement(statement)
                .parameters(parameters)
                .build();

        return ddb.executeStatement(request);
    }

    private static void processResults(ExecuteStatementResponse
 executeStatementResult) {
        System.out.println("ExecuteStatement successful: " +
 executeStatementResult.toString());
    }
}
```

- For API details, see ExecuteStatement in *AWS SDK for Java 2.x API Reference*.

Query for TTL items

The following code example shows how to query for TTL items.

SDK for Java 2.x

Query Filtered Expression to gather TTL items in a DynamoDB table.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

        // Get current time in epoch second format (comparing against expiry
 attribute)
        final long currentTime = System.currentTimeMillis() / 1000;
```

```
        // A string that contains conditions that DynamoDB applies after the Query
 operation, but before the data is returned to you.
        final String keyConditionExpression = "#pk = :pk";

        // The condition that specifies the key values for items to be retrieved by
 the Query action.
        final String filterExpression = "#ea > :ea";
        final Map<String, String> expressionAttributeNames = ImmutableMap.of(
                "#pk", "primaryKey",
                "#ea", "expireAt");
        final Map<String, AttributeValue> expressionAttributeValues =
 ImmutableMap.of(
                ":pk", AttributeValue.builder().s(primaryKey).build(),
                ":ea",
 AttributeValue.builder().s(String.valueOf(currentTime)).build()
        );

        final QueryRequest request = QueryRequest.builder()
                .tableName(tableName)
                .keyConditionExpression(keyConditionExpression)
                .filterExpression(filterExpression)
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .build();
        try (DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build()) {
            final QueryResponse response = ddb.query(request);
            System.out.println(tableName + " Query operation with TTL successful.
 Request id is "
                    + response.responseMetadata().requestId());
            // Print the items that are not expired
            for (Map<String, AttributeValue> item : response.items()) {
                System.out.println(item.toString());
            }
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.exit(0);
```

- For API details, see [Query](#) in *AWS SDK for Java 2.x API Reference*.

Update an item's TTL

The following code example shows how to update an item's TTL.

SDK for Java 2.x

Update TTL on an existing DynamoDB item in a table.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

        // Get current time in epoch second format
        final long currentTime = System.currentTimeMillis() / 1000;
        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (90 * 24 * 60 * 60);
        // An expression that defines one or more attributes to be updated, the
 action to be performed on them, and new values for them.
        final String updateExpression = "SET updatedAt=:c, expireAt=:e";

        final ImmutableMap<String, AttributeValue> keyMap =
                ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
                        "sortKey", AttributeValue.fromS(sortKey));
        final Map<String, AttributeValue> expressionAttributeValues =
 ImmutableMap.of(
                ":c",
 AttributeValue.builder().s(String.valueOf(currentTime)).build(),
                ":e", AttributeValue.builder().s(String.valueOf(expireDate)).build()
        );

        final UpdateItemRequest request = UpdateItemRequest.builder()
```

```
                .tableName(tableName)
                .key(keyMap)
                .updateExpression(updateExpression)
                .expressionAttributeValues(expressionAttributeValues)
                .build();
        try (DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build()) {
            final UpdateItemResponse response = ddb.updateItem(request);
            System.out.println(tableName + " UpdateItem operation with TTL
successful. Request id is "
                    + response.responseMetadata().requestId());
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.exit(0);
```

- For API details, see [UpdateItem](#) in *AWS SDK for Java 2.x API Reference*.

Serverless examples

Invoke a Lambda function from a DynamoDB trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving records from a DynamoDB stream. The function retrieves the DynamoDB payload and logs the record contents.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming a DynamoDB event with Lambda using Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import
 com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " + GSON.toJson(record.getDynamodb()));
    }
}
```

Reporting batch item failures for Lambda functions with a DynamoDB trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from a DynamoDB stream. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Reporting DynamoDB batch item failures with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 Serializable> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
 {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
```

```
}
```

Amazon EC2 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon EC2.

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

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

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

Get started

Hello Amazon EC2

The following code examples show how to get started using Amazon EC2.

SDK for Java 2.x

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

```
    public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
        try {
            DescribeSecurityGroupsRequest request =
 DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId)
                .build();

            // Use a paginator.
```

```
            DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
            listGroups.stream()
                .flatMap(r -> r.securityGroups().stream())
                .forEach(group -> System.out
                    .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeSecurityGroups in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

AllocateAddress

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

SDK for Java 2.x

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

```
    public static String allocateAddress(Ec2Client ec2) {
        try {
            AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
```

```
                    .domain(DomainType.VPC)
                    .build();

            AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
            return allocateResponse.allocationId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see AllocateAddress in *AWS SDK for Java 2.x API Reference*.

`AssociateAddress`

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

SDK for Java 2.x

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

```
    public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
        try {
            AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
                    .instanceId(instanceId)
                    .allocationId(allocationId)
                    .build();

            AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
            return associateResponse.associationId();
```

```
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see [AssociateAddress](#) in *AWS SDK for Java 2.x API Reference*.

AuthorizeSecurityGroupIngress

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

SDK for Java 2.x

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

```
    public static String createSecurityGroup(Ec2Client ec2, String groupName, String
 groupDesc, String vpcId,
            String myIpAddress) {
        try {
            CreateSecurityGroupRequest createRequest =
 CreateSecurityGroupRequest.builder()
                    .groupName(groupName)
                    .description(groupDesc)
                    .vpcId(vpcId)
                    .build();

            CreateSecurityGroupResponse resp =
 ec2.createSecurityGroup(createRequest);
            IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/0")
                    .build();

            IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
```

```
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

            IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

            AuthorizeSecurityGroupIngressRequest authRequest =
 AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

            ec2.authorizeSecurityGroupIngress(authRequest);
            System.out.println("Successfully added ingress policy to security group
 " + groupName);
            return resp.groupId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see AuthorizeSecurityGroupIngress in *AWS SDK for Java 2.x API Reference*.

`CreateKeyPair`

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

SDK for Java 2.x

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

```
    public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
 {
        try {
            CreateKeyPairRequest request = CreateKeyPairRequest.builder()
                    .keyName(keyName)
                    .build();

            CreateKeyPairResponse response = ec2.createKeyPair(request);
            String content = response.keyMaterial();
            BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
            writer.write(content);
            writer.close();
            System.out.println("Successfully created key pair named " + keyName);

        } catch (Ec2Exception | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see CreateKeyPair in *AWS SDK for Java 2.x API Reference*.

`CreateSecurityGroup`

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

SDK for Java 2.x

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

```
    public static String createSecurityGroup(Ec2Client ec2, String groupName, String
 groupDesc, String vpcId,
            String myIpAddress) {
        try {
            CreateSecurityGroupRequest createRequest =
 CreateSecurityGroupRequest.builder()
```

```
                    .groupName(groupName)
                    .description(groupDesc)
                    .vpcId(vpcId)
                    .build();

            CreateSecurityGroupResponse resp =
 ec2.createSecurityGroup(createRequest);
            IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/0")
                    .build();

            IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

            IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

            AuthorizeSecurityGroupIngressRequest authRequest =
 AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

            ec2.authorizeSecurityGroupIngress(authRequest);
            System.out.println("Successfully added ingress policy to security group
 " + groupName);
            return resp.groupId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateSecurityGroup in *AWS SDK for Java 2.x API Reference*.

DeleteKeyPair

The following code example shows how to use DeleteKeyPair.

SDK for Java 2.x

Note

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

```
    public static void deleteKeys(Ec2Client ec2, String keyPair) {
        try {
            DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
                    .keyName(keyPair)
                    .build();

            ec2.deleteKeyPair(request);
            System.out.println("Successfully deleted key pair named " + keyPair);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteKeyPair in *AWS SDK for Java 2.x API Reference*.

DeleteSecurityGroup

The following code example shows how to use DeleteSecurityGroup.

SDK for Java 2.x

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

```
    public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
        try {
            DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
                    .groupId(groupId)
                    .build();

            ec2.deleteSecurityGroup(request);
            System.out.println("Successfully deleted security group with Id " +
groupId);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteSecurityGroup in *AWS SDK for Java 2.x API Reference*.

`DescribeInstanceTypes`

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

SDK for Java 2.x

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

```
    // Get a list of instance types.
    public static String getInstanceTypes(Ec2Client ec2) {
        String instanceType;
        try {
            DescribeInstanceTypesRequest typesRequest =
 DescribeInstanceTypesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstanceTypesResponse response =
 ec2.describeInstanceTypes(typesRequest);
            List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
            for (InstanceTypeInfo type : instanceTypes) {
                System.out.println("The memory information of this type is " +
 type.memoryInfo().sizeInMiB());
                System.out.println("Network information is " +
 type.networkInfo().toString());
                System.out.println("Instance type is " +
 type.instanceType().toString());
                instanceType = type.instanceType().toString();
                if (instanceType.compareTo("t2.2xlarge")  == 0){
                    return instanceType;
                }
            }

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see DescribeInstanceTypes in *AWS SDK for Java 2.x API Reference*.

DescribeInstances

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

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeInstances {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Instances(ec2);
        ec2.close();
    }

    public static void describeEC2Instances(Ec2Client ec2) {
        try {
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstancesIterable instancesIterable =
 ec2.describeInstancesPaginator(request);
            instancesIterable.stream()
```

```
                .flatMap(r -> r.reservations().stream())
                .flatMap(reservation -> reservation.instances().stream())
                .forEach(instance -> {
                    System.out.println("Instance Id is " + instance.instanceId());
                    System.out.println("Image id is " + instance.imageId());
                    System.out.println("Instance type is " +
 instance.instanceType());
                    System.out.println("Instance state name is " +
 instance.state().name());
                    System.out.println("Monitoring information is " +
 instance.monitoring().state());
                });

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorCode());
            System.exit(1);
        }
    }
}
```

- For API details, see [DescribeInstances](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeKeyPairs`

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

SDK for Java 2.x

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

```
    public static void describeKeys(Ec2Client ec2) {
        try {
            DescribeKeyPairsResponse response = ec2.describeKeyPairs();
            response.keyPairs().forEach(keyPair -> System.out.printf(
                    "Found key pair with name %s " +
                            "and fingerprint %s",
                    keyPair.keyName(),
```

```
                    keyPair.keyFingerprint()));

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeKeyPairs in *AWS SDK for Java 2.x API Reference*.

DescribeSecurityGroups

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

SDK for Java 2.x

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

```
    public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
        try {
            DescribeSecurityGroupsRequest request =
 DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId)
                .build();

            // Use a paginator.
            DescribeSecurityGroupsIterable listGroups =
 ec2.describeSecurityGroupsPaginator(request);
            listGroups.stream()
                .flatMap(r -> r.securityGroups().stream())
                .forEach(group -> System.out
                    .println(" Group id: " +group.groupId() + " group name = " +
 group.groupName()));

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for Java 2.x API Reference*.

DisassociateAddress

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

SDK for Java 2.x

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

```
    public static void disassociateAddress(Ec2Client ec2, String associationId) {
        try {
            DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
                    .associationId(associationId)
                    .build();

            ec2.disassociateAddress(addressRequest);
            System.out.println("You successfully disassociated the address!");

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DisassociateAddress](#) in *AWS SDK for Java 2.x API Reference*.

ReleaseAddress

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples).

```
    public static void releaseEC2Address(Ec2Client ec2, String allocId) {
        try {
            ReleaseAddressRequest request = ReleaseAddressRequest.builder()
                    .allocationId(allocId)
                    .build();

            ec2.releaseAddress(request);
            System.out.println("Successfully released Elastic IP address " +
 allocId);
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [ReleaseAddress](https://docs.aws.amazon.com/goto/SdkForJavaV2/ec2-2016-11-15/ReleaseAddress) in *AWS SDK for Java 2.x API Reference*.

`RunInstances`

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
```

```
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This code example requires an AMI value. You can learn more about this value
 * by reading this documentation topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/AMIs.html
 */
public class CreateInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <name> <amiId>

                Where:
                   name - An instance name value that you can obtain from the AWS
 Console (for example, ami-xxxxxx5c8b987b1a0).\s
                   amiId - An Amazon Machine Image (AMI) value that you can obtain
 from the AWS Console (for example, i-xxxxxx2734106d0ab).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String amiId = args[1];
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
```

```
            .build();

        String instanceId = createEC2Instance(ec2, name, amiId);
        System.out.println("The Amazon EC2 Instance ID is " + instanceId);
        ec2.close();
    }

    public static String createEC2Instance(Ec2Client ec2, String name, String amiId)
 {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .imageId(amiId)
            .instanceType(InstanceType.T1_MICRO)
            .maxCount(1)
            .minCount(1)
            .build();

        // Use a waiter to wait until the instance is running.
        System.out.println("Going to start an EC2 instance using a waiter");
        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceIdVal = response.instances().get(0).instanceId();
        ec2.waiter().waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal));
        Tag tag = Tag.builder()
            .key("Name")
            .value(name)
            .build();

        CreateTagsRequest tagRequest = CreateTagsRequest.builder()
            .resources(instanceIdVal)
            .tags(tag)
            .build();

        try {
            ec2.createTags(tagRequest);
            System.out.printf("Successfully started EC2 Instance %s based on AMI
 %s", instanceIdVal, amiId);
            return instanceIdVal;

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

```
}
```

- For API details, see RunInstances in *AWS SDK for Java 2.x API Reference*.

StartInstances

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

SDK for Java 2.x

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

```
    public static void startInstance(Ec2Client ec2, String instanceId) {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                .overrideConfiguration(b -> b.maxAttempts(100))
                .client(ec2)
                .build();

        StartInstancesRequest request = StartInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to run. This
 will take a few minutes.");
        ec2.startInstances(request);
        DescribeInstancesRequest instanceRequest =
 DescribeInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
 ec2Waiter.waitUntilInstanceRunning(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
    }
```

- For API details, see [StartInstances](#) in *AWS SDK for Java 2.x API Reference*.

StopInstances

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

SDK for Java 2.x

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

```
    public static void stopInstance(Ec2Client ec2, String instanceId) {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                .overrideConfiguration(b -> b.maxAttempts(100))
                .client(ec2)
                .build();
        StopInstancesRequest request = StopInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
 will take a few minutes.");
        ec2.stopInstances(request);
        DescribeInstancesRequest instanceRequest =
 DescribeInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
 ec2Waiter.waitUntilInstanceStopped(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully stopped instance " + instanceId);
    }
```

- For API details, see [StopInstances](#) in *AWS SDK for Java 2.x API Reference*.

TerminateInstances

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

SDK for Java 2.x

Note

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

```
    public static void terminateEC2(Ec2Client ec2, String instanceId) {
        try {
            Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                    .overrideConfiguration(b -> b.maxAttempts(100))
                    .client(ec2)
                    .build();

            TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build();

            System.out.println("Use an Ec2Waiter to wait for the instance to
 terminate. This will take a few minutes.");
            ec2.terminateInstances(ti);
            DescribeInstancesRequest instanceRequest =
 DescribeInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build();

            WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
                    .waitUntilInstanceTerminated(instanceRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Successfully started instance " + instanceId);
            System.out.println(instanceId + " is terminated!");

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see TerminateInstances in *AWS SDK for Java 2.x API Reference*.

Scenarios

Build and manage a resilient service

The following code example shows how to create a load-balanced web service that returns book, movie, and song recommendations. The example shows how the service responds to failures, and how to restructure the service for more resilience when failures occur.

- Use an Amazon EC2 Auto Scaling group to create Amazon Elastic Compute Cloud (Amazon EC2) instances based on a launch template and to keep the number of instances in a specified range.
- Handle and distribute HTTP requests with Elastic Load Balancing.
- Monitor the health of instances in an Auto Scaling group and forward requests only to healthy instances.
- Run a Python web server on each EC2 instance to handle HTTP requests. The web server responds with recommendations and health checks.
- Simulate a recommendation service with an Amazon DynamoDB table.
- Control web server response to requests and health checks by updating AWS Systems Manager parameters.

SDK for Java 2.x

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

Run the interactive scenario at a command prompt.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
```

```
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
 {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
 a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
```

```
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
                To keep things tidy and to avoid unwanted charges on your account,
 we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
 might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
 autoScaler, Database database)
```

```
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
 InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to create
 several AWS resources
                        to set up a load-balanced web service endpoint and explore
 some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
 provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances that
 each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
 across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
 targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
 tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
 instance starts.
                This script starts a Python web server defined in the `server.py`
 script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
 and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
 production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

                The template also defines an IAM policy that each instance uses to
 assume a role that grants
                permissions to access the DynamoDB recommendation table and Systems
 Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
 startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
 instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
 autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
 starts, it listens for
                HTTP requests. You can see these instances in the console or
 continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
 The target group
                defines how the load balancer connects to instances. The load
 balancer provides a
                single endpoint where clients connect and dispatches requests to
 instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
 subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
 vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
 lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
 targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
 that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
 IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();
```

```
                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
 ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
 your default VPC must
                            allow access from this computer. You can either add it
 automatically from this
                            example or add it yourself using the AWS Management
 Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
 groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
 your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
 String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
 browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
 balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
 are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
 balancer.");
        }
```

```
        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
```

```
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                       """);
       demoChoices(loadBalancer);

       System.out.println(
               """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
                       While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                       """);
       paramHelper.put(paramHelper.failureResponse, "static");

       System.out.println("""
               Now, sending a GET request to the load balancer endpoint returns a
static response.
               The service still reports as healthy because health checks are still
shallow.
               """);
       demoChoices(loadBalancer);

       System.out.println("Let's reinstate the recommendation service.");
       paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

       System.out.println("""
               Let's also substitute bad credentials for one of the instances in
the target group so that it can't
               access the DynamoDB recommendation table. We will get an instance id
value.
               """);

       LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
       AutoScaler autoScaler = new AutoScaler();

       // Create a new instance profile based on badCredsProfileName.
       templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
       String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
       System.out.println("The bad instance id values used for this demo is " +
badInstanceId);
```

```
        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
 profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
 returns either a recommendation or a static response,
                        depending on which instance is selected by the load
 balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
 check tests whether
                the web service can access the DynamoDB table that it depends on for
 recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
 Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
 instance health, because it
                risks accidental termination of all instances in the Auto Scaling
 group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
 when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
 credentials
                is unhealthy. Note that it might take a minute or two for the load
 balancer to detect the unhealthy
```

```
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
```

```
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
 one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
 loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
 HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
 loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
 InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
```

```
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }
```

```
    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Create a class that wraps Auto Scaling and Amazon EC2 actions.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }
```

```
    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
 TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


 getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
 newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
 iamInstanceProfile =
 software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
 is a valid IAM Instance Profile
```

```
                                                            // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
```

```
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
```

```
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
 getInstanceProfileRequest =
 software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
 getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
 rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
```

```
                System.out.println("Detached and deleted policy " +
 attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
 name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


 getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
```

```
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
 DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
 securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
 cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }
```

```
                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be 
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to 
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String 
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest = 
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

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

```
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
 autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
 zonesRequest =
 software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
 getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
 zonesResponse.availabilityZones().stream()

 .map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
 LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
 CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
 autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
 software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
```

```
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
 DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
 getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
 DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
```

```
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
```

```
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Create a class that wraps Elastic Load Balancing actions.

```
public class LoadBalancer {
```

```
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
```

```
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();
```

```
        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();
```

```
        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();
```

```
            System.out.println("Waiting for Load Balancer " + lbName + " to become
 available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
 loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .defaultActions(action)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Create a class that uses DynamoDB to simulate a recommendation service.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
```

```
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");
```

```
            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws IOException
 {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Create a class that wraps Systems Manager actions.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener

- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Get started with instances

The following code example shows how to:

- Create a key pair and security group.
- Select an Amazon Machine Image (AMI) and compatible instance type, then create an instance.
- Stop and restart the instance.
- Associate an Elastic IP address with your instance.
- Connect to your instance with SSH, then clean up resources.

SDK for Java 2.x

Note

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

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets more information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI’s
 * architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an
 * instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
 * 13. Displays SSH connection info for the instance.
 * 14. Disassociates and deletes the Elastic IP address.
 * 15. Terminates the instance and waits for it to terminate.
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
public class EC2Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
```

```
        final String usage = """

                Usage:
                   <keyName> <fileName> <groupName> <groupDesc> <vpcId>

                Where:
                   keyName -  A key pair name (for example, TestKeyPair).\s
                   fileName -  A file name where the key information is written to.
\s
                   groupName - The name of the security group.\s
                   groupDesc - The description of the security group.\s
                   vpcId - A VPC Id value. You can get this value from the AWS
 Management Console.\s
                   myIpAddress - The IP address of your development machine.\s

                """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyName = args[0];
        String fileName = args[1];
        String groupName = args[2];
        String groupDesc = args[3];
        String vpcId = args[4];
        String myIpAddress = args[5];

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

        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("1. Create an RSA key pair and save the private key
 material as a .pem file.");
        createKeyPair(ec2, keyName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List key pairs.");
        describeKeys(ec2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a security group.");
        String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId,
 myIpAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Display security group info for the newly created
 security group.");
        describeSecurityGroups(ec2, groupId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one
 with amzn2 in the name.");
        String instanceId = getParaValues(ssmClient);
        System.out.println("The instance Id is " + instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Get more information about an amzn2 image.");
        String amiValue = describeImage(ec2, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of instance types.");
        String instanceType = getInstanceTypes(ec2);
        System.out.println("The instance type is " + instanceType);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an instance.");
        String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
 amiValue);
```

```
        System.out.println("The instance Id is " + newInstanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Display information about the running instance. ");
        String ipAddress = describeEC2Instances(ec2, newInstanceId);
        System.out.println("You can SSH to the instance using this command:");
        System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Stop the instance and use a waiter.");
        stopInstance(ec2, newInstanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Start the instance and use a waiter.");
        startInstance(ec2, newInstanceId);
        ipAddress = describeEC2Instances(ec2, newInstanceId);
        System.out.println("You can SSH to the instance using this command:");
        System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Allocate an Elastic IP address and associate it with
 the instance.");
        String allocationId = allocateAddress(ec2);
        System.out.println("The allocation Id value is " + allocationId);
        String associationId = associateAddress(ec2, newInstanceId, allocationId);
        System.out.println("The associate Id value is " + associationId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Describe the instance again.");
        ipAddress = describeEC2Instances(ec2, newInstanceId);
        System.out.println("You can SSH to the instance using this command:");
        System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Disassociate and release the Elastic IP address.");
        disassociateAddress(ec2, associationId);
        releaseEC2Address(ec2, allocationId);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("15. Terminate the instance and use a waiter.");
        terminateEC2(ec2, newInstanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the security group.");
        deleteEC2SecGroup(ec2, groupId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Delete the key.");
        deleteKeys(ec2, keyName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("You successfully completed the Amazon EC2 scenario.");
        System.out.println(DASHES);
        ec2.close();
    }

    public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
        try {
            DeleteSecurityGroupRequest request =
 DeleteSecurityGroupRequest.builder()
                    .groupId(groupId)
                    .build();

            ec2.deleteSecurityGroup(request);
            System.out.println("Successfully deleted security group with Id " +
 groupId);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void terminateEC2(Ec2Client ec2, String instanceId) {
        try {
            Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                    .overrideConfiguration(b -> b.maxAttempts(100))
                    .client(ec2)
```

```
                    .build();

            TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build();

            System.out.println("Use an Ec2Waiter to wait for the instance to
 terminate. This will take a few minutes.");
            ec2.terminateInstances(ti);
            DescribeInstancesRequest instanceRequest =
 DescribeInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build();

            WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
                    .waitUntilInstanceTerminated(instanceRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Successfully started instance " + instanceId);
            System.out.println(instanceId + " is terminated!");

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteKeys(Ec2Client ec2, String keyPair) {
        try {
            DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
                    .keyName(keyPair)
                    .build();

            ec2.deleteKeyPair(request);
            System.out.println("Successfully deleted key pair named " + keyPair);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void releaseEC2Address(Ec2Client ec2, String allocId) {
        try {
            ReleaseAddressRequest request = ReleaseAddressRequest.builder()
```

```
                    .allocationId(allocId)
                    .build();

            ec2.releaseAddress(request);
            System.out.println("Successfully released Elastic IP address " +
allocId);
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disassociateAddress(Ec2Client ec2, String associationId) {
        try {
            DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
                    .associationId(associationId)
                    .build();

            ec2.disassociateAddress(addressRequest);
            System.out.println("You successfully disassociated the address!");

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
        try {
            AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
                    .instanceId(instanceId)
                    .allocationId(allocationId)
                    .build();

            AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
            return associateResponse.associationId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
        return "";
    }

    public static String allocateAddress(Ec2Client ec2) {
        try {
            AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
                    .domain(DomainType.VPC)
                    .build();

            AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
            return allocateResponse.allocationId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void startInstance(Ec2Client ec2, String instanceId) {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                .overrideConfiguration(b -> b.maxAttempts(100))
                .client(ec2)
                .build();

        StartInstancesRequest request = StartInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
        ec2.startInstances(request);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
```

```
    }

    public static void stopInstance(Ec2Client ec2, String instanceId) {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                .overrideConfiguration(b -> b.maxAttempts(100))
                .client(ec2)
                .build();
        StopInstancesRequest request = StopInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
        ec2.stopInstances(request);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully stopped instance " + instanceId);
    }

    public static String describeEC2Instances(Ec2Client ec2, String newInstanceId) {
        try {
            String pubAddress = "";
            boolean isRunning = false;
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                    .instanceIds(newInstanceId)
                    .build();

            while (!isRunning) {
                DescribeInstancesResponse response = ec2.describeInstances(request);
                String state =
response.reservations().get(0).instances().get(0).state().name().name();
                if (state.compareTo("RUNNING") == 0) {
                    System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
                    System.out.println(
                            "Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
                    System.out.println(
```

```
                            "Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
                    pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
                    System.out.println("Instance address is " + pubAddress);
                    isRunning = true;
                }
            }
            return pubAddress;
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName,
            String amiId) {
        try {
            RunInstancesRequest runRequest = RunInstancesRequest.builder()
                    .instanceType(instanceType)
                    .keyName(keyName)
                    .securityGroups(groupName)
                    .maxCount(1)
                    .minCount(1)
                    .imageId(amiId)
                    .build();

            System.out.println("Going to start an EC2 instance using a waiter");
            RunInstancesResponse response = ec2.runInstances(runRequest);
            String instanceIdVal = response.instances().get(0).instanceId();
            ec2.waiter().waitUntilInstanceRunning(r ->
r.instanceIds(instanceIdVal));
            System.out.println("Successfully started EC2 instance " + instanceIdVal
+ " based on AMI " + amiId);
            return instanceIdVal;

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

```
    // Get a list of instance types.
    public static String getInstanceTypes(Ec2Client ec2) {
        String instanceType;
        try {
            DescribeInstanceTypesRequest typesRequest =
 DescribeInstanceTypesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstanceTypesResponse response =
 ec2.describeInstanceTypes(typesRequest);
            List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
            for (InstanceTypeInfo type : instanceTypes) {
                System.out.println("The memory information of this type is " +
 type.memoryInfo().sizeInMiB());
                System.out.println("Network information is " +
 type.networkInfo().toString());
                System.out.println("Instance type is " +
 type.instanceType().toString());
                instanceType = type.instanceType().toString();
                if (instanceType.compareTo("t2.2xlarge")  == 0){
                    return instanceType;
                }
            }

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Display the Description field that corresponds to the instance Id value.
    public static String describeImage(Ec2Client ec2, String instanceId) {
        try {
            DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
                    .imageIds(instanceId)
                    .build();

            DescribeImagesResponse response = ec2.describeImages(imagesRequest);
            System.out.println("The description of the first image is " +
 response.images().get(0).description());
```

```
            System.out.println("The name of the first image is " +
response.images().get(0).name());

            // Return the image Id value.
            return response.images().get(0).imageId();

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Get the Id value of an instance with amzn2 in the name.
    public static String getParaValues(SsmClient ssmClient) {
        try {
            GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
                    .path("/aws/service/ami-amazon-linux-latest")
                    .build();

            GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
            for
(software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse response :
responses) {
                System.out.println("Test " + response.nextToken());
                List<Parameter> parameterList = response.parameters();
                for (Parameter para : parameterList) {
                    System.out.println("The name of the para is: " + para.name());
                    System.out.println("The type of the para is: " + para.type());
                    if (filterName(para.name())) {
                        return para.value();
                    }
                }
            }

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

```
    // Return true if the name has amzn2 in it. For example:
    // /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
    private static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
        try {
            DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId)
                .build();

            // Use a paginator.
            DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
            listGroups.stream()
                .flatMap(r -> r.securityGroups().stream())
                .forEach(group -> System.out
                    .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
            String myIpAddress) {
        try {
            CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
                    .groupName(groupName)
                    .description(groupDesc)
                    .vpcId(vpcId)
                    .build();

            CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
            IpRange ipRange = IpRange.builder()
```

```
                    .cidrIp(myIpAddress + "/0")
                    .build();

            IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

            IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

            AuthorizeSecurityGroupIngressRequest authRequest =
 AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

            ec2.authorizeSecurityGroupIngress(authRequest);
            System.out.println("Successfully added ingress policy to security group
 " + groupName);
            return resp.groupId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void describeKeys(Ec2Client ec2) {
        try {
            DescribeKeyPairsResponse response = ec2.describeKeyPairs();
            response.keyPairs().forEach(keyPair -> System.out.printf(
                    "Found key pair with name %s " +
                            "and fingerprint %s",
                    keyPair.keyName(),
                    keyPair.keyFingerprint()));
```

```
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
 {
        try {
            CreateKeyPairRequest request = CreateKeyPairRequest.builder()
                    .keyName(keyName)
                    .build();

            CreateKeyPairResponse response = ec2.createKeyPair(request);
            String content = response.keyMaterial();
            BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
            writer.write(content);
            writer.close();
            System.out.println("Successfully created key pair named " + keyName);

        } catch (Ec2Exception | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances

- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Amazon ECR examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon ECR.

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

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

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

Get started

Hello Amazon ECR

The following code examples show how to get started using Amazon ECR.

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;

public class HelloECR {

    public static void main(String[] args) {
        final String usage = """
            Usage:    <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }
    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();

        ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
        imagesIterable.stream()
            .flatMap(r -> r.imageIds().stream())
            .forEach(image -> System.out.println("The docker image tag is: "
 +image.imageTag()));
    }
}
```

- For API details, see [listImages](#) in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

CreateRepository

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

SDK for Java 2.x

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

```
    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
 string if the operation failed.
     * @throws IllegalArgumentException     If repository name is invalid.
     * @throws RuntimeException             if an error occurs while creating the
 repository.
     */
    public String createECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
 empty");
        }

        CreateRepositoryRequest request = CreateRepositoryRequest.builder()
            .repositoryName(repoName)
            .build();
```

```
        CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
successfully.");
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }
```

- For API details, see CreateRepository in *AWS SDK for Java 2.x API Reference*.

`DeleteRepository`

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

SDK for Java 2.x

Note

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

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
     * @throws RuntimeException if an unexpected error occurs during the deletion
 process.
     */
    public void deleteECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
 empty");
        }

        DeleteRepositoryRequest repositoryRequest =
 DeleteRepositoryRequest.builder()
            .force(true)
            .repositoryName(repoName)
            .build();

        CompletableFuture<DeleteRepositoryResponse> response =
 getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " + repoName +
 " repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
```

```
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }
```

- For API details, see [DeleteRepository](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeImages`

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

SDK for Java 2.x

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

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
 information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
 exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
```

```
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- For API details, see [DescribeImages](#) in *AWS SDK for Java 2.x API Reference*.

DescribeRepositories

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

SDK for Java 2.x

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

```
    /**
```

```
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     * @throws EcrException        if there is an error retrieving the repository
 information.
     * @throws CompletionException if the asynchronous operation completes
 exceptionally.
     */
    public void getRepositoryURI(String repoName) {
        DescribeRepositoriesRequest request = DescribeRepositoriesRequest.builder()
            .repositoryNames(repoName)
            .build();

        CompletableFuture<DescribeRepositoriesResponse> response =
 getAsyncClient().describeRepositories(request);
        response.whenComplete((describeRepositoriesResponse, ex) -> {
            if (ex != null) {
                Throwable cause = ex.getCause();
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
 asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
 describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
 repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
 name.");
                    }
                } else {
                    System.err.println("No response received from
 describeRepositories.");
```

```
                }
            }
        });
        response.join();
    }
```

- For API details, see [DescribeRepositories](#) in *AWS SDK for Java 2.x API Reference*.

`GetAuthorizationToken`

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

SDK for Java 2.x

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

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
 (ECR).
     * This method makes an asynchronous call to the ECR client to retrieve the
 authorization token.
     * If the operation is successful, the method prints the token to the console.
     * If an exception occurs, the method handles the exception and prints the error
 message.
     *
     * @throws EcrException     if there is an error retrieving the authorization
 token from ECR.
     * @throws RuntimeException if there is an unexpected error during the
 operation.
     */
    public void getAuthToken() {
        CompletableFuture<GetAuthorizationTokenResponse> response =
 getAsyncClient().getAuthorizationToken();
        response.whenComplete((authorizationTokenResponse, ex) -> {
            if (authorizationTokenResponse != null) {
                AuthorizationData authorizationData =
 authorizationTokenResponse.authorizationData().get(0);
```

```
                String token = authorizationData.authorizationToken();
                if (!token.isEmpty()) {
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }
```

- For API details, see [GetAuthorizationToken](#) in *AWS SDK for Java 2.x API Reference*.

GetRepositoryPolicy

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

SDK for Java 2.x

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

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     * @throws EcrException if an AWS error occurs while getting the repository
policy.
     */
    public String getRepoPolicy(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
```

```
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy retrieved successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }
```

- For API details, see GetRepositoryPolicy in *AWS SDK for Java 2.x API Reference*.

PushImageCmd

The following code example shows how to use PushImageCmd.

SDK for Java 2.x

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

```
    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a few
seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
response.authorizationData().get(0).authorizationToken();
                String decodedToken = new String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
repoData.repositoryUri() + ":latest", imageName).exec();
                try {
```

```
getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }
```

- For API details, see [PushImageCmd](#) in *AWS SDK for Java 2.x API Reference*.

SetRepositoryPolicy

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

SDK for Java 2.x

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

```
    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole  the IAM role to be granted access to the repository.
     * @throws RepositoryPolicyNotFoundException if the repository policy does not
 exist.
     * @throws EcrException                      if there is an unexpected error
 setting the repository policy.
     */
    public void setRepoPolicy(String repoName, String iamRole) {
        /*
```

```
         This example policy document grants the specified AWS principal the
permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the specified
principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
            {
             "Version" : "2012-10-17",
             "Statement" : [ {
               "Sid" : "new statement",
               "Effect" : "Allow",
               "Principal" : {
                 "AWS" : "%s"
               },
               "Action" : "ecr:BatchGetImage"
             } ]
           }
            """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
       SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .policyText(policyDocument)
           .build();

       CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy set successfully.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof RepositoryPolicyNotFoundException) {
                   throw (RepositoryPolicyNotFoundException) cause;
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               }
           }
       });
```

```
        response.join();
    }
```

- For API details, see SetRepositoryPolicy in *AWS SDK for Java 2.x API Reference*.

StartLifecyclePolicyPreview

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

SDK for Java 2.x

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

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
```

```
                            throw (EcrException) cause;
                        } else {
                            throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                        }
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                    }
                } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                    System.out.println("Image is present in the repository.");
                } else {
                    System.out.println("Image is not present in the repository.");
                }
            });

            // Wait for the CompletableFuture to complete.
            response.join();
        }
```

- For API details, see StartLifecyclePolicyPreview in *AWS SDK for Java 2.x API Reference*.

Scenarios

Learn Amazon ECR core operations

The following code example shows how to:

- Create an Amazon ECR repository.
- Set repository policies.
- Retrieve repository URIs.
- Get Amazon ECR authorization tokens.
- Set lifecycle policies for Amazon ECR repositories.
- Push a Docker image to an Amazon ECR repository.
- Verify the existence of an image in an Amazon ECR repository.
- List Amazon ECR repositories for your account and get details about them.
- Delete Amazon ECR repositories.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ecr#code-examples).

Run an interactive scenario demonstrating Amazon ECR features.

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;

import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example requires an IAM Role that has permissions to interact with
 the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text. Without
 a local image,
 * this Java program will not successfully run. For more information including how
 to create the local
 * image, see:
 *
 * /getting_started_scenarios/ecr_scenario/README
 *
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) {
        final String usage = """
```

```
            Usage: <iamRoleARN> <accountId>

            Where:
               iamRoleARN - The IAM role ARN that has the necessary permissions to
access and manage the Amazon ECR repository.
               accountId - Your AWS account number.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        ECRActions ecrActions = new ECRActions();
        String iamRole = args[0];
        String accountId = args[1];
        String localImageName;

        Scanner scanner = new Scanner(System.in);
        System.out.println("""
             The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
container registry
             service provided by AWS. It allows developers and organizations to
securely
             store, manage, and deploy Docker container images.
             ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
             from building and testing to production deployment.\s

             The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides a
set of methods to
             programmatically interact with the Amazon ECR service. This allows
developers to
             automate the storage, retrieval, and management of container images as
part of their application
             deployment pipelines. With ECR, teams can focus on building and
deploying their
             applications without having to worry about the underlying
infrastructure required to
             host and manage a container registry.

            This scenario walks you through how to perform key operations for this
service.
            Let's get started...
```

```
            You have two choices:
            1 - Run the entire program.
            2 - Delete an existing Amazon ECR repository named echo-text (created
 from a previous execution of
            this program that did not complete).
            """);

        while (true) {
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("1")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else if (input.trim().equalsIgnoreCase("2")) {
                String repoName = "echo-text";
                ecrActions.deleteECRRepository(repoName);
                return;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("""
           1. Create an ECR repository.

           The first task is to ensure we have a local Docker image named echo-
text.
           If this image exists, then an Amazon ECR repository is created.

           An ECR repository is a private Docker container repository provided
           by Amazon Web Services (AWS). It is a managed service that makes it easy
           to store, manage, and deploy Docker container images.\s
           """ );

        // Ensure that a local docker image named echo-text exists.
        boolean doesExist = ecrActions.isEchoTextImagePresent();
        String repoName;
        if (!doesExist){
            System.out.println("The local image named echo-text does not exist");
```

```
            return;
        } else {
            localImageName = "echo-text";
            repoName = "echo-text";
        }

        try {
            String repoArn = ecrActions.createECRRepository(repoName);
            System.out.println("The ARN of the ECR repository is " + repoArn);

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
 " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function is
 crucial for maintaining
        the security and integrity of your container images. The repository policy
 allows you to
        define specific rules and restrictions for accessing and managing the images
 stored within your ECR
        repository.
        """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setRepoPolicy(repoName, iamRole);

        } catch (RepositoryPolicyNotFoundException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
```

```
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """);
        waitForInputToContinue(scanner);
        try {
            String policyText = ecrActions.getRepoPolicy(repoName);
            System.out.println("Policy Text:");
            System.out.println(policyText);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is responsible
for obtaining a
        valid authorization token, which is required to authenticate your requests
to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
```

```
        ECR repository, such as pushing, pulling, or managing your Docker images.

        """);
        waitForInputToContinue(scanner);
        try {
             ecrActions.getAuthToken();

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the authorization
 token: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to deploy a
 container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
 (EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
 image URI,
        which includes the ECR repository URI. This allows the container runtime to
 pull the
        correct container image from the ECR repository.
       """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.getRepositoryURI(repoName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;

        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the URI: " +
 e.getMessage());
            return;
```

```
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            6. Set an ECR Lifecycle Policy.

            An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
 stored in your ECR repositories.
            These policies allow you to automatically remove old or unused Docker
 images from your repositories,
            freeing up storage space and reducing costs.

            This example policy helps to maintain the size and efficiency of the
 container registry
            by automatically removing older and potentially unused images, ensuring
 that the
            storage is optimized and the registry remains up-to-date.
            """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setLifeCyclePolicy(repoName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while setting the lifecycle
 policy: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
 repository.
        It sets up the Docker client by connecting to the local Docker host using
 the default port.
        It then retrieves the authorization token for the ECR repository by making a
 call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
 which is used to authenticate
```

```
        the Docker client when pushing the image. Finally, the method tags the 
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR 
repository using the Docker client.
        If the push operation is successful, the method prints a message indicating 
that the image was pushed to ECR.
        """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.pushDockerImage(repoName, localImageName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while pushing a local Docker image 
to Amazon ECR: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("8. Verify if the image is in the ECR Repository.");
        waitForInputToContinue(scanner);
        try {
            ecrActions.verifyImage(repoName, localImageName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("9. As an optional step, you can interact with the image 
in Amazon ECR by using the CLI.");
        System.out.println("Would you like to view instructions on how to use the 
CLI to run the image? (y/n)");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String instructions = """
```

```
            1. Authenticate with ECR - Before you can pull the image from Amazon
 ECR, you need to authenticate with the registry. You can do this using the AWS CLI:

                aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:

               aws ecr describe-images --repository-name %s --image-ids imageTag=%s

            3. Run the Docker container and view the output using this command:

               docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
            """;

            instructions = String.format(instructions, accountId, repoName,
 localImageName, accountId, repoName, localImageName);
            System.out.println(instructions);
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("10. Delete the ECR Repository.");
        System.out.println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository and
 have Amazon ECR delete all of its contents
        on your behalf.
        """);
        System.out.println("Would you like to delete the Amazon ECR Repository? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the AWS ECR resources.");

            try {
                ecrActions.deleteECRRepository(repoName);

            } catch (EcrException e) {
                System.err.println("An ECR exception occurred: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
```

```
                System.err.println("An error occurred while deleting the Docker
 image: " + e.getMessage());
                e.printStackTrace();
                return;
            }
        }

        System.out.println(DASHES);
        System.out.println("This concludes the Amazon ECR SDK scenario");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
           }
        }
    }
}
```

A wrapper class for Amazon ECR SDK methods.

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
import software.amazon.awssdk.services.ecr.model.Repository;
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

public class ECRActions {
    private static EcrAsyncClient ecrClient;

    private static DockerClient dockerClient;

    private static Logger logger = LoggerFactory.getLogger(ECRActions.class);

    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
 string if the operation failed.
```

```
     * @throws IllegalArgumentException     If repository name is invalid.
     * @throws RuntimeException             if an error occurs while creating the
 repository.
     */
    public String createECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
 empty");
        }

        CreateRepositoryRequest request = CreateRepositoryRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<CreateRepositoryResponse> response =
 getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
 successfully.");
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
 ("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
 moving on...");
                    DescribeRepositoriesRequest describeRequest =
 DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
 getAsyncClient().describeRepositories(describeRequest).join();
                    return describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
```

```
            }
        }
    }

    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
     * @throws RuntimeException if an unexpected error occurs during the deletion
 process.
     */
    public void deleteECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
 empty");
        }

        DeleteRepositoryRequest repositoryRequest =
 DeleteRepositoryRequest.builder()
            .force(true)
            .repositoryName(repoName)
            .build();

        CompletableFuture<DeleteRepositoryResponse> response =
 getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " + repoName +
 " repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
```

```
    }



    private static DockerClient getDockerClient() {
        String osName = System.getProperty("os.name");
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
 default port.
            DockerCmdExecFactory dockerCmdExecFactory = new
 NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
            dockerClient =
 DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory).
        } else {
            dockerClient = DockerClientBuilder.getInstance().build();
        }
        return dockerClient;
    }

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static EcrAsyncClient getAsyncClient() {

        /*
         The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
         and it is designed to provide a high-performance, asynchronous HTTP client
 for interacting with AWS services.
         It uses the Netty framework to handle the underlying network communication
 and the Java NIO API to
         provide a non-blocking, event-driven approach to HTTP requests and
 responses.
         */
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(50)  // Adjust as needed.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();
```

```
        ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
 call attempt timeout.
            .build();

        if (ecrClient == null) {
            ecrClient = EcrAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ecrClient;
    }

    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    public void setLifeCyclePolicy(String repoName) {
        /*
           This policy helps to maintain the size and efficiency of the container
 registry
           by automatically removing older and potentially unused images,
           ensuring that the storage is optimized and the registry remains up-to-
date.
         */
        String polText = """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
```

```
                    },
                    "action": {
                        "type": "expire"
                    }
                }
            ]
            }
            """;

        StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
StartLifecyclePolicyPreviewRequest.builder()
            .lifecyclePolicyText(polText)
            .repositoryName(repoName)
            .build();

        CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
        response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
            if (lifecyclePolicyPreviewResponse != null) {
                System.out.println("Lifecycle policy preview started
successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });
        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
```

```
     * @throws CompletionException      if the asynchronous operation completes
 exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     * @throws EcrException        if there is an error retrieving the repository
 information.
```

```
     * @throws CompletionException if the asynchronous operation completes
exceptionally.
     */
    public void getRepositoryURI(String repoName) {
        DescribeRepositoriesRequest request = DescribeRepositoriesRequest.builder()
            .repositoryNames(repoName)
            .build();

        CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
        response.whenComplete((describeRepositoriesResponse, ex) -> {
            if (ex != null) {
                Throwable cause = ex.getCause();
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }
```

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
 (ECR).
     * This method makes an asynchronous call to the ECR client to retrieve the
 authorization token.
     * If the operation is successful, the method prints the token to the console.
     * If an exception occurs, the method handles the exception and prints the error
 message.
     *
     * @throws EcrException     if there is an error retrieving the authorization
 token from ECR.
     * @throws RuntimeException if there is an unexpected error during the
 operation.
     */
    public void getAuthToken() {
        CompletableFuture<GetAuthorizationTokenResponse> response =
 getAsyncClient().getAuthorizationToken();
        response.whenComplete((authorizationTokenResponse, ex) -> {
            if (authorizationTokenResponse != null) {
                AuthorizationData authorizationData =
 authorizationTokenResponse.authorizationData().get(0);
                String token = authorizationData.authorizationToken();
                if (!token.isEmpty()) {
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
 ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
 exception
                }
            }
        });
        response.join();
    }

    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
```

```
     * @throws EcrException if an AWS error occurs while getting the repository
 policy.
     */
    public String getRepoPolicy(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
 empty");
        }

        GetRepositoryPolicyRequest getRepositoryPolicyRequest =
 GetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<GetRepositoryPolicyResponse> response =
 getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy retrieved successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
 ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }

    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole  the IAM role to be granted access to the repository.
     * @throws RepositoryPolicyNotFoundException if the repository policy does not
 exist.
     * @throws EcrException                      if there is an unexpected error
 setting the repository policy.
     */
```

```
    public void setRepoPolicy(String repoName, String iamRole) {
        /*
          This example policy document grants the specified AWS principal the
 permission to perform the
          `ecr:BatchGetImage` action. This policy is designed to allow the specified
 principal
          to retrieve Docker images from the ECR repository.
         */
        String policyDocumentTemplate = """
             {
              "Version" : "2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "%s"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }
             """;

        String policyDocument = String.format(policyDocumentTemplate, iamRole);
        SetRepositoryPolicyRequest setRepositoryPolicyRequest =
 SetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .policyText(policyDocument)
            .build();

        CompletableFuture<SetRepositoryPolicyResponse> response =
 getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy set successfully.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof RepositoryPolicyNotFoundException) {
                    throw (RepositoryPolicyNotFoundException) cause;
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
```

```
            }
        });
        response.join();
    }

    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
 repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a few
 seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
 getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
 response.authorizationData().get(0).authorizationToken();
                String decodedToken = new String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
```

```
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }

    // Make sure local image echo-text exists.
    public boolean isEchoTextImagePresent() {
        try {
            List<Image> images = getDockerClient().listImagesCmd().exec();
            boolean helloWorldFound = false;
            for (Image image : images) {
                String[] repoTags = image.getRepoTags();
                if (repoTags != null) {
                    for (String tag : repoTags) {
                        if (tag.startsWith("echo-text")) {
                            System.out.println(tag);
                            helloWorldFound = true;
                        }
                    }
                }
            }
            if (helloWorldFound) {
                System.out.println("The local image named echo-text exists.");
                return true;
            } else {
                System.out.println("The local image named echo-text does not
 exist.");
                return false;
            }
        } catch (DockerClientException ex) {
            logger.error("ERROR: " + ex.getMessage());
            return false;
```

```
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Amazon ECS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon ECS.

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

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

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

Actions

CreateCluster

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ecs#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandConfiguration;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandLogging;
import software.amazon.awssdk.services.ecs.model.ClusterConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateClusterResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.CreateClusterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCluster {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <clusterName>\s

                Where:
                   clusterName - The name of the ECS cluster to create.
                """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        String clusterArn = createGivenCluster(ecsClient, clusterName);
        System.out.println("The cluster ARN is " + clusterArn);
        ecsClient.close();
    }

    public static String createGivenCluster(EcsClient ecsClient, String clusterName)
 {
        try {
            ExecuteCommandConfiguration commandConfiguration =
 ExecuteCommandConfiguration.builder()
                    .logging(ExecuteCommandLogging.DEFAULT)
                    .build();

            ClusterConfiguration clusterConfiguration =
 ClusterConfiguration.builder()
                    .executeCommandConfiguration(commandConfiguration)
                    .build();

            CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
                    .clusterName(clusterName)
                    .configuration(clusterConfiguration)
                    .build();

            CreateClusterResponse response =
 ecsClient.createCluster(clusterRequest);
            return response.cluster().clusterArn();

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

- For API details, see CreateCluster in *AWS SDK for Java 2.x API Reference*.

CreateService

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.AwsVpcConfiguration;
import software.amazon.awssdk.services.ecs.model.NetworkConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateServiceRequest;
import software.amazon.awssdk.services.ecs.model.LaunchType;
import software.amazon.awssdk.services.ecs.model.CreateServiceResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateService {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                  <clusterName> <serviceName> <securityGroups>
 <subnets> <taskDefinition>

                                Where:
```

```
                                 clusterName - The name of the ECS cluster.
                                 serviceName - The name of the ECS service to
create.
                                 securityGroups - The name of the security group.
                                 subnets - The name of the subnet.
                                 taskDefinition - The name of the task definition.
                               """;

                if (args.length != 5) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String clusterName = args[0];
                String serviceName = args[1];
                String securityGroups = args[2];
                String subnets = args[3];
                String taskDefinition = args[4];
                Region region = Region.US_EAST_1;
                EcsClient ecsClient = EcsClient.builder()
                                .region(region)
                                .build();

                String serviceArn = createNewService(ecsClient, clusterName,
serviceName, securityGroups, subnets,
                                taskDefinition);
                System.out.println("The ARN of the service is " + serviceArn);
                ecsClient.close();
        }

        public static String createNewService(EcsClient ecsClient,
                        String clusterName,
                        String serviceName,
                        String securityGroups,
                        String subnets,
                        String taskDefinition) {

                try {
                        AwsVpcConfiguration vpcConfiguration =
AwsVpcConfiguration.builder()
                                        .securityGroups(securityGroups)
                                        .subnets(subnets)
                                        .build();
```

```
                        NetworkConfiguration configuration =
 NetworkConfiguration.builder()
                                        .awsvpcConfiguration(vpcConfiguration)
                                        .build();

                        CreateServiceRequest serviceRequest =
 CreateServiceRequest.builder()
                                        .cluster(clusterName)
                                        .networkConfiguration(configuration)
                                        .desiredCount(1)
                                        .launchType(LaunchType.FARGATE)
                                        .serviceName(serviceName)
                                        .taskDefinition(taskDefinition)
                                        .build();

                        CreateServiceResponse response =
 ecsClient.createService(serviceRequest);
                        return response.service().serviceArn();

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

- For API details, see CreateService in *AWS SDK for Java 2.x API Reference*.

`DeleteService`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DeleteServiceRequest;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterName> <serviceArn>\s

                Where:
                  clusterName - The name of the ECS cluster.
                  serviceArn - The ARN of the ECS service.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        deleteSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }
```

```
    public static void deleteSpecificService(EcsClient ecsClient, String
 clusterName, String serviceArn) {
        try {
            DeleteServiceRequest serviceRequest = DeleteServiceRequest.builder()
                    .cluster(clusterName)
                    .service(serviceArn)
                    .build();

            ecsClient.deleteService(serviceRequest);
            System.out.println("The Service was successfully deleted");

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

- For API details, see [DeleteService](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeClusters`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeClustersRequest;
import software.amazon.awssdk.services.ecs.model.DescribeClustersResponse;
import software.amazon.awssdk.services.ecs.model.Cluster;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeClusters {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterArn> \s

                Where:
                  clusterArn - The ARN of the ECS cluster to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        descCluster(ecsClient, clusterArn);
    }

    public static void descCluster(EcsClient ecsClient, String clusterArn) {
        try {
            DescribeClustersRequest clustersRequest =
 DescribeClustersRequest.builder()
                    .clusters(clusterArn)
                    .build();

            DescribeClustersResponse response =
 ecsClient.describeClusters(clustersRequest);
            List<Cluster> clusters = response.clusters();
            for (Cluster cluster : clusters) {
                System.out.println("The cluster name is " + cluster.clusterName());
```

```
            }

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

- For API details, see [DescribeClusters](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeTasks`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeTasksRequest;
import software.amazon.awssdk.services.ecs.model.DescribeTasksResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.Task;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTaskDefinitions {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                  <clusterArn> <taskId>\s

                Where:
                  clusterArn - The ARN of an ECS cluster.
                  taskId - The task Id value.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        String taskId = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        getAllTasks(ecsClient, clusterArn, taskId);
        ecsClient.close();
    }

    public static void getAllTasks(EcsClient ecsClient, String clusterArn, String
 taskId) {
        try {
            DescribeTasksRequest tasksRequest = DescribeTasksRequest.builder()
                    .cluster(clusterArn)
                    .tasks(taskId)
                    .build();

            DescribeTasksResponse response = ecsClient.describeTasks(tasksRequest);
            List<Task> tasks = response.tasks();
            for (Task task : tasks) {
                System.out.println("The task ARN is " + task.taskDefinitionArn());
            }

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

```
    }
}
```

- For API details, see DescribeTasks in *AWS SDK for Java 2.x API Reference*.

ListClusters

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

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ListClustersResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        listAllClusters(ecsClient);
        ecsClient.close();
```

```
    }

    public static void listAllClusters(EcsClient ecsClient) {
        try {
            ListClustersResponse response = ecsClient.listClusters();
            List<String> clusters = response.clusterArns();
            for (String cluster : clusters) {
                System.out.println("The cluster arn is " + cluster);
            }

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

- For API details, see ListClusters in *AWS SDK for Java 2.x API Reference*.

`UpdateService`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.UpdateServiceRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class UpdateService {

    public static void main(String[] args) {

        final String usage = """

                Usage:
                   <clusterName> <serviceArn>\s

                Where:
                  clusterName - The cluster name.
                  serviceArn - The service ARN value.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        updateSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }

    public static void updateSpecificService(EcsClient ecsClient, String
 clusterName, String serviceArn) {
        try {
            UpdateServiceRequest serviceRequest = UpdateServiceRequest.builder()
                    .cluster(clusterName)
                    .service(serviceArn)
                    .desiredCount(0)
                    .build();

            ecsClient.updateService(serviceRequest);
```

```
            System.out.println("The service was modified");

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

- For API details, see UpdateService in *AWS SDK for Java 2.x API Reference*.

Elastic Load Balancing - Version 2 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Elastic Load Balancing - Version 2.

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

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

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

Get started

Hello Elastic Load Balancing

The following code examples show how to get started using Elastic Load Balancing.

SDK for Java 2.x

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

```
public class HelloLoadBalancer {

        public static void main(String[] args) {
                ElasticLoadBalancingV2Client loadBalancingV2Client =
 ElasticLoadBalancingV2Client.builder()
                                .region(Region.US_EAST_1)
                                .build();

                DescribeLoadBalancersResponse loadBalancersResponse =
 loadBalancingV2Client
                                .describeLoadBalancers(r -> r.pageSize(10));
                List<LoadBalancer> loadBalancerList =
 loadBalancersResponse.loadBalancers();
                for (LoadBalancer lb : loadBalancerList)
                        System.out.println("Load Balancer DNS name = " +
 lb.dnsName());
        }
}
```

- For API details, see DescribeLoadBalancers in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

`CreateListener`

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

SDK for Java 2.x

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

```
    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);
```

```
            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .defaultActions(action)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- For API details, see CreateListener in *AWS SDK for Java 2.x API Reference*.

`CreateLoadBalancer`

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/elbv2#code-examples).

```
    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
```

```
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
 loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .defaultActions(action)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- For API details, see [CreateLoadBalancer] in *AWS SDK for Java 2.x API Reference*.

`CreateTargetGroup`

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

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/elasticloadbalancing#code-examples).

```
    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
 targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
 CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
 getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
 targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
 targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
 targetGroupArn);
        return targetGroupArn;
    }
```

- For API details, see CreateTargetGroup in *AWS SDK for Java 2.x API Reference*.

DeleteLoadBalancer

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

SDK for Java 2.x

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

```
    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
 getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
 DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
 builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
 loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }
```

- For API details, see DeleteLoadBalancer in *AWS SDK for Java 2.x API Reference*.

DeleteTargetGroup

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

SDK for Java 2.x

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

```
    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
 describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
 builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

- For API details, see DeleteTargetGroup in *AWS SDK for Java 2.x API Reference*.

DescribeTargetHealth

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

SDK for Java 2.x

Note

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

```
    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }
```

- For API details, see DescribeTargetHealth in *AWS SDK for Java 2.x API Reference*.

Scenarios

Build and manage a resilient service

The following code example shows how to create a load-balanced web service that returns book, movie, and song recommendations. The example shows how the service responds to failures, and how to restructure the service for more resilience when failures occur.

- Use an Amazon EC2 Auto Scaling group to create Amazon Elastic Compute Cloud (Amazon EC2) instances based on a launch template and to keep the number of instances in a specified range.

- Handle and distribute HTTP requests with Elastic Load Balancing.
- Monitor the health of instances in an Auto Scaling group and forward requests only to healthy instances.
- Run a Python web server on each EC2 instance to handle HTTP requests. The web server responds with recommendations and health checks.
- Simulate a recommendation service with an Amazon DynamoDB table.
- Control web server response to requests and health checks by updating AWS Systems Manager parameters.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples).

Run the interactive scenario at a command prompt.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";
```

```
    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
 {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
 a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
```

```
                To keep things tidy and to avoid unwanted charges on your account,
 we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
 might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
 autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
 InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
```

```
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to create
 several AWS resources
                        to set up a load-balanced web service endpoint and explore
 some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
 provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances that
 each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
 across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
 targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
 tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
 instance starts.
                This script starts a Python web server defined in the `server.py`
 script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
 and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
 production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

                The template also defines an IAM policy that each instance uses to
 assume a role that grants
```

```
                permissions to access the DynamoDB recommendation table and Systems
Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
The target group
                defines how the load balancer connects to instances. The load
balancer provides a
```

```
               single endpoint where clients connect and dispatches requests to
instances in the group.
               """);

       String vpcId = autoScaler.getDefaultVPC();
       List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
       System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
       String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
       String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
       autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
       System.out.println("Verifying access to the load balancer endpoint...");
       boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
       if (!wasSuccessul) {
           System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
           CloseableHttpClient httpClient = HttpClients.createDefault();

           // Create an HTTP GET request to "http://checkip.amazonaws.com"
           HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
           try {
               // Execute the request and get the response
               HttpResponse response = httpClient.execute(httpGet);

               // Read the response content.
               String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

               // Print the public IP address.
               System.out.println("Public IP Address: " + ipAddress);
               GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
               if (!groupInfo.isPortOpen()) {
                   System.out.println("""
                           For this example to work, the default security group for
your default VPC must
                           allow access from this computer. You can either add it
automatically from this
                           example or add it yourself using the AWS Management
Console.
                           """);
```

```
                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);
```

```
        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
 different parts of the system
                        to create situations where the web service fails, and shows
 how using a resilient
                        architecture can keep the web service running in spite of
 these failures.

                        At the start, the load balancer endpoint returns
 recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
 recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
 named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
 set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer endpoint
 returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
 don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
 the web service can return a static response.
                       While this is not a perfect solution, it presents the
 customer with a somewhat better experience than failure.
                       """);
```

```
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns a
 static response.
                The service still reports as healthy because health checks are still
 shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
 the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance id
 value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
 badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
 badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
 profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
 returns either a recommendation or a static response,
                        depending on which instance is selected by the load
 balancer.
                        """);
```

```
        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
 check tests whether
                the web service can access the DynamoDB table that it depends on for
 recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
 Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
 instance health, because it
                risks accidental termination of all instances in the Auto Scaling
 group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
 when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
 credentials
                is unhealthy. Note that it might take a minute or two for the load
 balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
 returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
 scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
 new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);
```

```
        System.out.println("""
                Even while the instance is terminating and the new instance is
 starting, sending a GET
                request to the web service continues to get a successful
 recommendation response because
                the load balancer routes requests to the healthy instances. After
 the replacement instance
                starts and reports as healthy, it is included in the load balancing
 rotation.
                Note that terminating and replacing an instance typically takes
 several minutes, during which time you
                can see the changing health check status until the new instance is
 running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
 all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
 InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
 one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
```

```
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
 loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
 HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
 loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
 InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
 targets:\n");
```

```
                        List<TargetHealthDescription> health =
 loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
 check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Create a class that wraps Auto Scaling and Amazon EC2 actions.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
```

```
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
```

```
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
 TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


 getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
 newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
 iamInstanceProfile =
 software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
 is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
 ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
 'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
```

```
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with profile
 %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
 waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
 getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
 informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
```

```
                        Collections.singletonList("cd / && sudo python3 server.py 
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " + 
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest = 
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.", 
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest 
getInstanceProfileRequest = 
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response = 
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest = 
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
```

```
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
```

```
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


 getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
 DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();
```

```
            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
```

```
        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

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

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
```

```
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
```

```
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
 DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
 getAutoScalingClient().describeAutoScalingGroups(request);
```

```
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
 DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
```

```
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
```

```
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Create a class that wraps Elastic Load Balancing actions.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
```

```
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
```

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

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
 describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
 builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
 InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
```

```
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
```

```
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
```

```
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .defaultActions(action)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Create a class that uses DynamoDB to simulate a recommendation service.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
```

```
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
```

```
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
```

```
    public void populateTable(String fileName, String tableName) throws IOException
 {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Create a class that wraps Systems Manager actions.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
```

```
            put(failureResponse, "none");
            put(healthCheck, "shallow");
        }

        public void put(String name, String value) {
            SsmClient ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();

            PutParameterRequest parameterRequest = PutParameterRequest.builder()
                    .name(name)
                    .value(value)
                    .overwrite(true)
                    .type("String")
                    .build();

            ssmClient.putParameter(parameterRequest);
            System.out.printf("Setting demo parameter %s to '%s'.", name, value);
        }
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones

- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

MediaStore examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with MediaStore.

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

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

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

`CreateContainer`

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/mediastore#code-examples).

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
```

```
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
 String containerName) {
        try {
            CreateContainerRequest containerRequest =
 CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
 mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
 containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
 containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [CreateContainer](#) in *AWS SDK for Java 2.x API Reference*.

DeleteContainer

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

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/mediastore#code-examples).

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
```

```
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
 String containerName) {
        try {
            CreateContainerRequest containerRequest =
 CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
 mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
 containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
 containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteContainer in *AWS SDK for Java 2.x API Reference*.

DeleteObject

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

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName>

                Where:
                   completePath - The path (including the container) of the item to
 delete.
                   containerName - The name of the container.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String completePath = args[0];
        String containerName = args[1];
        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));

        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        deleteMediaObject(mediaStoreData, completePath);
        mediaStoreData.close();
    }

    public static void deleteMediaObject(MediaStoreDataClient mediaStoreData, String
 completePath) {
        try {
            DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                    .path(completePath)
                    .build();

            mediaStoreData.deleteObject(deleteObjectRequest);

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

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        mediaStoreClient.close();
```

```
        return response.container().endpoint();
    }
}
```

- For API details, see [DeleteObject](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeContainer`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeContainer {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to describe.
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
 containerName));
        mediaStoreClient.close();
    }

    public static String checkContainer(MediaStoreClient mediaStoreClient, String
 containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
 DescribeContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            DescribeContainerResponse containerResponse =
 mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
 containerResponse.container().name());
            System.out.println("The container ARN is " +
 containerResponse.container().arn());
            return containerResponse.container().status().toString();

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

- For API details, see DescribeContainer in *AWS SDK for Java 2.x API Reference*.

GetObject

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

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName> <savePath>

                Where:
```

```
                  completePath - The path of the object in the container (for
example, Videos5/sampleVideo.mp4).
                  containerName - The name of the container.
                  savePath - The path on the local drive where the file is saved,
including the file name (for example, C:/AWS/myvid.mp4).
               """;

       if (args.length != 3) {
           System.out.println(usage);
           System.exit(1);
       }

       String completePath = args[0];
       String containerName = args[1];
       String savePath = args[2];

       Region region = Region.US_EAST_1;
       URI uri = new URI(getEndpoint(containerName));
       MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
               .endpointOverride(uri)
               .region(region)
               .build();

       getMediaObject(mediaStoreData, completePath, savePath);
       mediaStoreData.close();
   }

   public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
completePath, String savePath) {

       try {
           GetObjectRequest objectRequest = GetObjectRequest.builder()
                   .path(completePath)
                   .build();

           // Write out the data to a file.
           ResponseInputStream<GetObjectResponse> data =
mediaStoreData.getObject(objectRequest);
           byte[] buffer = new byte[data.available()];
           data.read(buffer);

           File targetFile = new File(savePath);
           OutputStream outStream = new FileOutputStream(targetFile);
           outStream.write(buffer);
```

```
            System.out.println("The data was written to " + savePath);

        } catch (MediaStoreDataException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- For API details, see GetObject in *AWS SDK for Java 2.x API Reference*.

ListContainers

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
            ListContainersResponse containersResponse =
 mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();
            for (Container container : containers) {
                System.out.println("Container name is " + container.name());
            }

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

- For API details, see ListContainers in *AWS SDK for Java 2.x API Reference*.

PutObject

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = """

                To run this example, supply the name of a container, a file location
 to use, and path in the container\s
```

```
                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
 filePath, String completePath) {
        try {
            File myFile = new File(filePath);
            RequestBody requestBody = RequestBody.fromFile(myFile);

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .path(completePath)
                    .contentType("video/mp4")
                    .build();

            PutObjectResponse response = mediaStoreData.putObject(objectRequest,
 requestBody);
            System.out.println("The saved object is " +
 response.storageClass().toString());

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

```
    public static String getEndpoint(String containerName) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- For API details, see PutObject in *AWS SDK for Java 2.x API Reference*.

OpenSearch Service examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with OpenSearch Service.

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

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

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

CreateDomain

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

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/opensearch#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.CreateDomainResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDomain {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <domainName>

                Where:
                    domainName - The name of the domain to create.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
                .region(region)
                .build();

        createNewDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void createNewDomain(OpenSearchClient searchClient, String
 domainName) {
        try {
            ClusterConfig clusterConfig = ClusterConfig.builder()
                    .dedicatedMasterEnabled(true)
                    .dedicatedMasterCount(3)
                    .dedicatedMasterType("t2.small.search")
                    .instanceType("t2.small.search")
                    .instanceCount(5)
                    .build();

            EBSOptions ebsOptions = EBSOptions.builder()
                    .ebsEnabled(true)
                    .volumeSize(10)
                    .volumeType(VolumeType.GP2)
                    .build();

            NodeToNodeEncryptionOptions encryptionOptions =
 NodeToNodeEncryptionOptions.builder()
                    .enabled(true)
                    .build();

            CreateDomainRequest domainRequest = CreateDomainRequest.builder()
                    .domainName(domainName)
                    .engineVersion("OpenSearch_1.0")
                    .clusterConfig(clusterConfig)
                    .ebsOptions(ebsOptions)
                    .nodeToNodeEncryptionOptions(encryptionOptions)
                    .build();
```

```
            System.out.println("Sending domain creation request...");
            CreateDomainResponse createResponse =
 searchClient.createDomain(domainRequest);
            System.out.println("Domain status is " +
 createResponse.domainStatus().toString());
            System.out.println("Domain Id is " +
 createResponse.domainStatus().domainId());

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

- For API details, see CreateDomain in *AWS SDK for Java 2.x API Reference*.

DeleteDomain

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

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDomain {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <domainName>

                Where:
                    domainName - The name of the domain to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
                .region(region)
                .build();

        deleteSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void deleteSpecificDomain(OpenSearchClient searchClient, String
 domainName) {
        try {
            DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
                    .domainName(domainName)
                    .build();

            searchClient.deleteDomain(domainRequest);
            System.out.println(domainName + " was successfully deleted.");

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

- For API details, see DeleteDomain in *AWS SDK for Java 2.x API Reference*.

ListDomainNames

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListDomainNames {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
                .region(region)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
        listAllDomains(searchClient);
```

```
        System.out.println("Done");
    }

    public static void listAllDomains(OpenSearchClient searchClient) {
        try {
            ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
                    .engineType("OpenSearch")
                    .build();

            ListDomainNamesResponse response =
 searchClient.listDomainNames(namesRequest);
            List<DomainInfo> domainInfoList = response.domainNames();
            for (DomainInfo domain : domainInfoList)
                System.out.println("Domain name is " + domain.domainName());

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

- For API details, see ListDomainNames in *AWS SDK for Java 2.x API Reference*.

`UpdateDomainConfig`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
```

```
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateDomain {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <domainName>

                Where:
                    domainName - The name of the domain to update.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
                .region(region)
                .build();

        updateSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void updateSpecificDomain(OpenSearchClient searchClient, String
 domainName) {
        try {
            ClusterConfig clusterConfig = ClusterConfig.builder()
                    .instanceCount(3)
                    .build();
```

```
            UpdateDomainConfigRequest updateDomainConfigRequest =
 UpdateDomainConfigRequest.builder()
                    .domainName(domainName)
                    .clusterConfig(clusterConfig)
                    .build();

            System.out.println("Sending domain update request...");
            UpdateDomainConfigResponse updateResponse =
 searchClient.updateDomainConfig(updateDomainConfigRequest);
            System.out.println("Domain update response from Amazon OpenSearch
 Service:");
            System.out.println(updateResponse.toString());

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

- For API details, see UpdateDomainConfig in *AWS SDK for Java 2.x API Reference*.

EventBridge examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with EventBridge.

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

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

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

Get started

Hello EventBridge

The following code examples show how to get started using EventBridge.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/eventbridge#code-examples).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        listBuses(eventBrClient);
        eventBrClient.close();
    }

    public static void listBuses(EventBridgeClient eventBrClient) {
        try {
            ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
                    .limit(10)
                    .build();

            ListEventBusesResponse response =
 eventBrClient.listEventBuses(busesRequest);
            List<EventBus> buses = response.eventBuses();
            for (EventBus bus : buses) {
                System.out.println("The name of the event bus is: " + bus.name());
```

```
                System.out.println("The ARN of the event bus is: " + bus.arn());
            }

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

- For API details, see ListEventBuses in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

`DeleteRule`

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

SDK for Java 2.x

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

```
    public static void deleteRuleByName(EventBridgeClient eventBrClient, String
 ruleName) {
        DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
                .name(ruleName)
                .build();

        eventBrClient.deleteRule(ruleRequest);
        System.out.println("Successfully deleted the rule");
```

```
    }
```

- For API details, see DeleteRule in *AWS SDK for Java 2.x API Reference*.

`DescribeRule`

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

SDK for Java 2.x

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

```
    public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
        try {
            DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                    .name(eventRuleName)
                    .build();

            DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
            System.out.println("The state of the rule is " +
response.stateAsString());

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

- For API details, see DescribeRule in *AWS SDK for Java 2.x API Reference*.

`DisableRule`

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

SDK for Java 2.x

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

Disable a rule by using its rule name.

```
    public static void changeRuleState(EventBridgeClient eventBrClient, String
 eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
            }

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

- For API details, see [DisableRule](#) in *AWS SDK for Java 2.x API Reference*.

EnableRule

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

SDK for Java 2.x

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

Enable a rule by using its rule name.

```
    public static void changeRuleState(EventBridgeClient eventBrClient, String
 eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
            }

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

- For API details, see EnableRule in *AWS SDK for Java 2.x API Reference*.

`ListRuleNamesByTarget`

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

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/eventbridge#code-examples).

List all of the rule names by using the target.

```
    public static void listTargetRules(EventBridgeClient eventBrClient, String
 topicArn) {
        ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
 ListRuleNamesByTargetRequest.builder()
                .targetArn(topicArn)
                .build();

        ListRuleNamesByTargetResponse response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
        List<String> rules = response.ruleNames();
        for (String rule : rules) {
            System.out.println("The rule name is " + rule);
        }
    }
```

- For API details, see ListRuleNamesByTarget in *AWS SDK for Java 2.x API Reference*.

`ListRules`

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

SDK for Java 2.x

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

Enable a rule by using its rule name.

```
    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
 rule.description());
                System.out.println("The rule state is : " + rule.stateAsString());
            }

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

- For API details, see ListRules in *AWS SDK for Java 2.x API Reference*.

ListTargetsByRule

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

SDK for Java 2.x

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

List all of the targets for a rule by using the rule name.

```
    public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
 {
```

```
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }
```

- For API details, see ListTargetsByRule in *AWS SDK for Java 2.x API Reference*.

PutEvents

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

SDK for Java 2.x

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

```
    public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
        String json = "{" +
                "\"UserEmail\": \"" + email + "\"," +
                "\"Message\": \"This event was generated by example code.\"," +
                "\"UtcTime\": \"Now.\"" +
                "}";

        PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
                .source("ExampleSource")
                .detail(json)
                .detailType("ExampleType")
                .build();
```

```
        PutEventsRequest eventsRequest = PutEventsRequest.builder()
                .entries(entry)
                .build();

        eventBrClient.putEvents(eventsRequest);
    }
```

- For API details, see PutEvents in *AWS SDK for Java 2.x API Reference*.

PutRule

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

SDK for Java 2.x

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

Create a scheduled rule.

```
    public static void createEBRule(EventBridgeClient eventBrClient, String
 ruleName, String cronExpression) {
        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .name(ruleName)
                    .eventBusName("default")
                    .scheduleExpression(cronExpression)
                    .state("ENABLED")
                    .description("A test rule that runs on a schedule created by the
 Java API")
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
 ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
```

Create a rule that triggers when an object is added to an Amazon Simple Storage Service bucket.

```
    // Create a new event rule that triggers when an Amazon S3 object is created in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
 String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
 ruleResponse.ruleArn());

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

- For API details, see PutRule in *AWS SDK for Java 2.x API Reference*.

PutTargets

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

SDK for Java 2.x

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

Add an Amazon SNS topic as a target for a rule.

```
    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
 ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
 target " + topicName + " for bucket "
                + bucketName + ".");
    }
```

Add an input transformer to a target for a rule.

```
    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
 eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see PutTargets in *AWS SDK for Java 2.x API Reference*.

`RemoveTargets`

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

SDK for Java 2.x

Note

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

Remove all of the targets for a rule by using the rule name.

```
    public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
 eventRuleName) {
        // First, get all targets that will be deleted.
        ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
                .rule(eventRuleName)
                .build();

        ListTargetsByRuleResponse response =
 eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
 RemoveTargetsRequest.builder()
                    .rule(eventRuleName)
                    .ids(myTarget.id())
                    .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }
```

- For API details, see RemoveTargets in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with rules and targets

The following code example shows how to:

- Create a rule and add a target to it.
- Enable and disable rules.
- List and update rules and targets.
- Send events, then clean up resources.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/eventbridge#code-examples).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
 * Amazon EventBridge.
 * 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
 * enabled.
 * 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 * 4. Lists rules on the event bus.
 * 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
 * lets the user subscribe to it.
 * 6. Adds a target to the rule that sends an email to the specified topic.
 * 7. Creates an EventBridge event that sends an email when an Amazon S3 object
 * is created.
 * 8. Lists Targets.
 * 9. Lists the rules for the same target.
 * 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
 * 11. Disables a specific rule.
 * 12. Checks and print the state of the rule.
 * 13. Adds a transform to the rule to change the text of the email.
 * 14. Enables a specific rule.
 * 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
 * 16. Updates the rule to be a custom rule pattern.
 * 17. Sending an event to trigger the rule.
 * 18. Cleans up resources.
```

```
 *
 */
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException, IOException
 {
        final String usage = """

                Usage:
                    <roleName> <bucketName> <topicName> <eventRuleName>

                Where:
                    roleName - The name of the role to create.
                    bucketName - The Amazon Simple Storage Service (Amazon S3)
 bucket name to create.
                    topicName - The name of the Amazon Simple Notification Service
 (Amazon SNS) topic to create.
                    eventRuleName - The Amazon EventBridge rule name to create.
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String polJSON = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        Scanner sc = new Scanner(System.in);
        String roleName = args[0];
        String bucketName = args[1];
        String topicName = args[2];
        String eventRuleName = args[3];

        Region region = Region.US_EAST_1;
```

```
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(regionGl)
                .build();

        SnsClient snsClient = SnsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EventBridge example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out
                .println("1. Create an AWS Identity and Access Management (IAM) role
 to use with Amazon EventBridge.");
        String roleArn = createIAMRole(iam, roleName, polJSON);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create an S3 bucket with EventBridge events
 enabled.");
        if (checkBucket(s3Client, bucketName)) {
            System.out.println("Bucket " + bucketName + " already exists. Ending
 this scenario.");
            System.exit(1);
        }

        createBucket(s3Client, bucketName);
        Thread.sleep(3000);
        setBucketNotification(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("3. Create a rule that triggers when an object is
 uploaded to Amazon S3.");
        Thread.sleep(10000);
        addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. List rules on the event bus.");
        listRules(eventBrClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new SNS topic for testing and let the user
 subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to the
 specified topic.");
        System.out.println("Enter your email to subscribe to the Amazon SNS
 topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println(
                "Use the link in the email you received to confirm your
 subscription. Then, press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email when
 an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
 eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 9. List the rules for the same target.");
```

```
        listTargetRules(eventBrClient, topicArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 10. Trigger the rule by uploading a file to the S3
 bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Disable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, false);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Check and print the state of the rule.");
        checkRule(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Add a transform to the rule to change the text of
 the email.");
        updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Enable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, true);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 15. Trigger the updated rule by uploading a file to the
 S3 bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 16. Update the rule to be a custom rule pattern.");
        updateToCustomRule(eventBrClient, eventRuleName);
```

```
        System.out.println("Updated event rule " + eventRuleName + " to use a custom
pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn, eventRuleName);
        System.out.println("Updated event target " + topicArn + ".");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient, SnsClient
snsClient, S3Client s3Client,
            IamClient iam, String topicArn, String eventRuleName, String bucketName,
String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }
```

```
    public static void deleteRole(IamClient iam, String roleName) {
        String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
        DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
                .policyArn(policyArn)
                .roleName(roleName)
                .build();

        iam.detachRolePolicy(policyRequest);
        System.out.println("Successfully detached policy " + policyArn + " from role
 " + roleName);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);
    }

    public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
        // Remove all the objects from the S3 bucket.
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
                .bucket(bucketName)
                .build();

        ListObjectsResponse res = s3Client.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

        for (S3Object myValue : objects) {
            toDelete.add(ObjectIdentifier.builder()
                    .key(myValue.key())
                    .build());
        }

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(Delete.builder()
                        .objects(toDelete).build())
                .build();

        s3Client.deleteObjects(dor);
```

```
        // Delete the S3 bucket.
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("You have deleted the bucket and the objects");
    }

    // Delete the SNS topic.
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

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

    public static void deleteRuleByName(EventBridgeClient eventBrClient, String
 ruleName) {
        DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
                .name(ruleName)
                .build();

        eventBrClient.deleteRule(ruleRequest);
        System.out.println("Successfully deleted the rule");
    }

    public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
 eventRuleName) {
        // First, get all targets that will be deleted.
        ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
                .rule(eventRuleName)
                .build();
```

```
        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                    .rule(eventRuleName)
                    .ids(myTarget.id())
                    .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }

    public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
        String json = "{" +
                "\"UserEmail\": \"" + email + "\"," +
                "\"Message\": \"This event was generated by example code.\"," +
                "\"UtcTime\": \"Now.\"" +
                "}";

        PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
                .source("ExampleSource")
                .detail(json)
                .detailType("ExampleType")
                .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
                .entries(entry)
                .build();

        eventBrClient.putEvents(eventsRequest);
    }

    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
```

```
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateToCustomRule(EventBridgeClient eventBrClient, String
 ruleName) {
        String customEventsPattern = "{" +
                "\"source\": [\"ExampleSource\"]," +
                "\"detail-type\": [\"ExampleType\"]" +
                "}";

        PutRuleRequest request = PutRuleRequest.builder()
                .name(ruleName)
                .description("Custom test rule")
                .eventPattern(customEventsPattern)
                .build();

        eventBrClient.putRule(request);
    }

    // Update an Amazon S3 object created rule with a transform on the target.
    public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
 topicArn, String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        Map<String, String> myMap = new HashMap<>();
        myMap.put("bucket", "$.detail.bucket.name");
```

```
        myMap.put("time", "$.time");

        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\"")
                .inputPathsMap(myMap)
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);

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

    public static void checkRule(EventBridgeClient eventBrClient, String
 eventRuleName) {
        try {
            DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                    .name(eventRuleName)
                    .build();

            DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
            System.out.println("The state of the rule is " +
 response.stateAsString());

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

```
    }

    public static void changeRuleState(EventBridgeClient eventBrClient, String
 eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
            }

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

    // Create and upload a file to an S3 bucket to trigger an event.
    public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
 throws IOException {
        // Create a unique file name.
        String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
 Date());
        String fileName = "TextFile" + fileSuffix + ".txt";

        File myFile = new File(fileName);
        FileWriter fw = new FileWriter(myFile.getAbsoluteFile());
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("This is a sample file for testing uploads.");
        bw.close();

        try {
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(fileName)
```

```
                    .build();

            s3Client.putObject(putOb, RequestBody.fromFile(myFile));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listTargetRules(EventBridgeClient eventBrClient, String
 topicArn) {
        ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
 ListRuleNamesByTargetRequest.builder()
                .targetArn(topicArn)
                .build();

        ListRuleNamesByTargetResponse response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
        List<String> rules = response.ruleNames();
        for (String rule : rules) {
            System.out.println("The rule name is " + rule);
        }
    }

    public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
 {
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
 eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }

    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
 ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
```

```
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
 target " + topicName + " for bucket "
                + bucketName + ".");
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String email)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                    + result.sdkHttpResponse().statusCode());

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

    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
```

```
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
rule.description());
                System.out.println("The rule state is : " + rule.stateAsString());
            }

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

    public static String createSnsTopic(SnsClient snsClient, String topicName) {
        String topicPolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Sid\": \"EventBridgePublishTopic\"," +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Resource\": \"*\"," +
                "\"Action\": \"sns:Publish\"" +
                "}]" +
                "}";

        Map<String, String> topicAttributes = new HashMap<>();
        topicAttributes.put("Policy", topicPolicy);
        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        System.out.println("Added topic " + topicName + " for email
subscriptions.");
        return response.topicArn();
```

```
    }

    // Create a new event rule that triggers when an Amazon S3 object is created in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
 String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
 ruleResponse.ruleArn());

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

    // Determine if the S3 bucket exists.
    public static Boolean checkBucket(S3Client s3Client, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.headBucket(headBucketRequest);
            return true;
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Set the S3 bucket notification configuration.
    public static void setBucketNotification(S3Client s3Client, String bucketName) {
        try {
            EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
                    .build();

            NotificationConfiguration configuration =
NotificationConfiguration.builder()
                    .eventBridgeConfiguration(eventBridgeConfiguration)
                    .build();

            PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
                    .build();

            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
```

```
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
 s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createIAMRole(IamClient iam, String rolename, String
 polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(polJSON)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            AttachRolePolicyRequest rolePolicyRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(rolename)
                    .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
                    .build();

            iam.attachRolePolicy(rolePolicyRequest);
            return response.role().arn();

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Forecast examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Forecast.

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

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

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

Actions

CreateDataset

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

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/forecast#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateDatasetRequest;
import software.amazon.awssdk.services.forecast.model.Schema;
import software.amazon.awssdk.services.forecast.model.SchemaAttribute;
import software.amazon.awssdk.services.forecast.model.CreateDatasetResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataSet {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <name>\s

                Where:
                    name - The name of the data set.\s
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        String myDataSetARN = createForecastDataSet(forecast, name);
        System.out.println("The ARN of the new data set is " + myDataSetARN);
        forecast.close();
    }

    public static String createForecastDataSet(ForecastClient forecast, String name)
 {
        try {
            Schema schema = Schema.builder()
                    .attributes(getSchema())
                    .build();

            CreateDatasetRequest datasetRequest = CreateDatasetRequest.builder()
                    .datasetName(name)
                    .domain("CUSTOM")
                    .datasetType("RELATED_TIME_SERIES")
                    .dataFrequency("D")
                    .schema(schema)
                    .build();

            CreateDatasetResponse response = forecast.createDataset(datasetRequest);
            return response.datasetArn();

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

        return "";
    }

    // Create a SchemaAttribute list required to create a data set.
    private static List<SchemaAttribute> getSchema() {
```

```
        List<SchemaAttribute> schemaList = new ArrayList<>();
        SchemaAttribute att1 = SchemaAttribute.builder()
                .attributeName("item_id")
                .attributeType("string")
                .build();

        SchemaAttribute att2 = SchemaAttribute.builder()
                .attributeName("timestamp")
                .attributeType("timestamp")
                .build();

        SchemaAttribute att3 = SchemaAttribute.builder()
                .attributeName("target_value")
                .attributeType("float")
                .build();

        // Push the SchemaAttribute objects to the List.
        schemaList.add(att1);
        schemaList.add(att2);
        schemaList.add(att3);
        return schemaList;
    }
}
```

- For API details, see CreateDataset in *AWS SDK for Java 2.x API Reference*.

`CreateForecast`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
```

```
import software.amazon.awssdk.services.forecast.model.CreateForecastRequest;
import software.amazon.awssdk.services.forecast.model.CreateForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateForecast {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <name> <predictorArn>\s

                Where:
                    name - The name of the forecast.\s
                    predictorArn - The arn of the predictor to use.\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String predictorArn = args[1];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        String forecastArn = createNewForecast(forecast, name, predictorArn);
        System.out.println("The ARN of the new forecast is " + forecastArn);
        forecast.close();
    }

    public static String createNewForecast(ForecastClient forecast, String name,
 String predictorArn) {
```

```
        try {
            CreateForecastRequest forecastRequest = CreateForecastRequest.builder()
                    .forecastName(name)
                    .predictorArn(predictorArn)
                    .build();

            CreateForecastResponse response =
 forecast.createForecast(forecastRequest);
            return response.forecastArn();

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

- For API details, see [CreateForecast](#) in *AWS SDK for Java 2.x API Reference*.

`DeleteDataset`

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

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <datasetARN>\s

                Where:
                    datasetARN - The ARN of the data set to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
 myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                    .datasetArn(myDataSetARN)
                    .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- For API details, see DeleteDataset in *AWS SDK for Java 2.x API Reference*.

DeleteForecast

The following code example shows how to use `DeleteForecast`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <datasetARN>\s

                Where:
```

```
                    datasetARN - The ARN of the data set to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
 myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                    .datasetArn(myDataSetARN)
                    .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteForecast in *AWS SDK for Java 2.x API Reference*.

DescribeForecast

The following code example shows how to use `DescribeForecast`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/forecast#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DescribeForecastRequest;
import software.amazon.awssdk.services.forecast.model.DescribeForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeForecast {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <forecastarn>\s

                Where:
                    forecastarn - The arn of the forecast (for example,
 "arn:aws:forecast:us-west-2:xxxxx322:forecast/my_forecast)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String forecastarn = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
```

```
                .region(region)
                .build();

        describe(forecast, forecastarn);
        forecast.close();
    }

    public static void describe(ForecastClient forecast, String forecastarn) {
        try {
            DescribeForecastRequest request = DescribeForecastRequest.builder()
                    .forecastArn(forecastarn)
                    .build();

            DescribeForecastResponse response = forecast.describeForecast(request);
            System.out.println("The name of the forecast is " +
 response.forecastName());

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DescribeForecast in *AWS SDK for Java 2.x API Reference*.

`ListDatasetGroups`

The following code example shows how to use `ListDatasetGroups`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DatasetGroupSummary;
```

```
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsRequest;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListDataSetGroups {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        listDataGroups(forecast);
        forecast.close();
    }

    public static void listDataGroups(ForecastClient forecast) {
        try {
            ListDatasetGroupsRequest group = ListDatasetGroupsRequest.builder()
                    .maxResults(10)
                    .build();

            ListDatasetGroupsResponse response = forecast.listDatasetGroups(group);
            List<DatasetGroupSummary> groups = response.datasetGroups();
            for (DatasetGroupSummary myGroup : groups) {
                System.out.println("The Data Set name is " +
 myGroup.datasetGroupName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListDatasetGroups in *AWS SDK for Java 2.x API Reference*.

ListForecasts

The following code example shows how to use `ListForecasts`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.ListForecastsResponse;
import software.amazon.awssdk.services.forecast.model.ListForecastsRequest;
import software.amazon.awssdk.services.forecast.model.ForecastSummary;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListForecasts {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        listAllForeCasts(forecast);
        forecast.close();
```

```
    }

    public static void listAllForeCasts(ForecastClient forecast) {
        try {
            ListForecastsRequest request = ListForecastsRequest.builder()
                    .maxResults(10)
                    .build();

            ListForecastsResponse response = forecast.listForecasts(request);
            List<ForecastSummary> forecasts = response.forecasts();
            for (ForecastSummary forecastSummary : forecasts) {
                System.out.println("The name of the forecast is " +
 forecastSummary.forecastName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListForecasts in *AWS SDK for Java 2.x API Reference*.

AWS Glue examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Glue.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello AWS Glue

The following code examples show how to get started using AWS Glue.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
                .maxResults(10)
                .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- For API details, see [ListJobs](#) in *AWS SDK for Java 2.x API Reference*.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

`CreateCrawler`

The following code example shows how to use `CreateCrawler`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.CreateCrawlerRequest;
import software.amazon.awssdk.services.glue.model.CrawlerTargets;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.S3Target;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCrawler {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <IAM> <s3Path> <cron> <dbName> <crawlerName>

                Where:
                    IAM - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                    s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                    cron - A cron expression used to specify the schedule  (i.e.,
cron(15 12 * * ? *).
                    dbName - The database name.\s
                    crawlerName - The name of the crawler.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        glueClient.close();
    }

    public static void createGlueCrawler(GlueClient glueClient,
            String iam,
            String s3Path,
            String cron,
            String dbName,
            String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                    .path(s3Path)
```

```
                    .build();

            // Add the S3Target to a list.
            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);

            CrawlerTargets targets = CrawlerTargets.builder()
                    .s3Targets(targetList)
                    .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                    .databaseName(dbName)
                    .name(crawlerName)
                    .description("Created by the AWS Glue Java API")
                    .targets(targets)
                    .role(iam)
                    .schedule(cron)
                    .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see CreateCrawler in *AWS SDK for Java 2.x API Reference*.

GetCrawler

The following code example shows how to use `GetCrawler`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetCrawlerRequest;
import software.amazon.awssdk.services.glue.model.GetCrawlerResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetCrawler {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <crawlerName>

                Where:
                    crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String crawlerName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        getSpecificCrawler(glueClient, crawlerName);
        glueClient.close();
```

```
    }

    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
 {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
            Instant createDate = response.crawler().creationTime();

            // Convert the Instant to readable date
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the Crawler is " + createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see GetCrawler in *AWS SDK for Java 2.x API Reference*.

`GetDatabase`

The following code example shows how to use `GetDatabase`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetDatabaseRequest;
import software.amazon.awssdk.services.glue.model.GetDatabaseResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetDatabase {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <databaseName>

                Where:
                    databaseName - The name of the database.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String databaseName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        getSpecificDatabase(glueClient, databaseName);
        glueClient.close();
```

```
    }

    public static void getSpecificDatabase(GlueClient glueClient, String
 databaseName) {
        try {
            GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
                    .name(databaseName)
                    .build();

            GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
            Instant createDate = response.database().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the database is " + createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see GetDatabase in *AWS SDK for Java 2.x API Reference*.

`GetTables`

The following code example shows how to use `GetTables`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetTableRequest;
import software.amazon.awssdk.services.glue.model.GetTableResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbName> <tableName>

                Where:
                    dbName - The database name.\s
                    tableName - The name of the table.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbName = args[0];
        String tableName = args[1];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();
```

```
        getGlueTable(glueClient, dbName, tableName);
        glueClient.close();
    }

    public static void getGlueTable(GlueClient glueClient, String dbName, String
 tableName) {
        try {
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .databaseName(dbName)
                    .name(tableName)
                    .build();

            GetTableResponse tableResponse = glueClient.getTable(tableRequest);
            Instant createDate = tableResponse.table().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the table is " + createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see GetTables in *AWS SDK for Java 2.x API Reference*.

StartCrawler

The following code example shows how to use `StartCrawler`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/glue#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.StartCrawlerRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartCrawler {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <crawlerName>

                Where:
                    crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String crawlerName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();
```

```
        startSpecificCrawler(glueClient, crawlerName);
        glueClient.close();
    }

    public static void startSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            glueClient.startCrawler(crawlerRequest);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see StartCrawler in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with crawlers and jobs

The following code example shows how to:

- Create a crawler that crawls a public Amazon S3 bucket and generates a database of CSV-formatted metadata.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create a job to extract CSV data from the S3 bucket, transform the data, and load JSON-formatted output into another S3 bucket.
- List information about job runs, view transformed data, and clean up resources.

For more information, see Tutorial: Getting started with AWS Glue Studio.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
 * 13. Delete a crawler.
 */

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
```

```
        final String usage = """

                Usage:
                    <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>\s

                Where:
                    iam - The ARN of the IAM role that has AWS Glue and S3
 permissions.\s
                    s3Path - The Amazon Simple Storage Service (Amazon S3) target
 that contains data (for example, CSV data).
                    cron - A cron expression used to specify the schedule  (i.e.,
 cron(15 12 * * ? *).
                    dbName - The database name.\s
                    crawlerName - The name of the crawler.\s
                    jobName - The name you assign to this job definition.
                    scriptLocation - The Amazon S3 path to a script that runs a job.
                    locationUri - The location of the database
                    bucketNameSc - The Amazon S3 bucket name used when creating a
 job
                    """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];
        String scriptLocation = args[6];
        String locationUri = args[7];
        String bucketNameSc = args[8];

        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Glue scenario.");
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("1. Create a database.");
        createDatabase(glueClient, dbName, locationUri);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
        createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get a crawler.");
        getSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        startSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        getSpecificDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait 5 min for the tables to become available");
        TimeUnit.MINUTES.sleep(5);
        System.out.println("6. Get tables.");
        String myTableName = getGlueTables(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        createJob(glueClient, jobName, iam, scriptLocation);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
```

```
        getAllJobs(glueClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        getJobRuns(glueClient, jobName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        deleteJob(glueClient, jobName);
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        deleteDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        deleteSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }

    public static void createDatabase(GlueClient glueClient, String dbName, String
 locationUri) {
        try {
            DatabaseInput input = DatabaseInput.builder()
                    .description("Built with the AWS SDK for Java V2")
                    .name(dbName)
                    .locationUri(locationUri)
                    .build();

            CreateDatabaseRequest request = CreateDatabaseRequest.builder()
                    .databaseInput(input)
                    .build();

            glueClient.createDatabase(request);
```

```
            System.out.println(dbName + " was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createGlueCrawler(GlueClient glueClient,
            String iam,
            String s3Path,
            String cron,
            String dbName,
            String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                    .path(s3Path)
                    .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                    .s3Targets(targetList)
                    .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                    .databaseName(dbName)
                    .name(crawlerName)
                    .description("Created by the AWS Glue Java API")
                    .targets(targets)
                    .role(iam)
                    .schedule(cron)
                    .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
 {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            boolean ready = false;
            while (!ready) {
                GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
                if (status.compareTo("READY") == 0) {
                    ready = true;
                }
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void startSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificDatabase(GlueClient glueClient, String
 databaseName) {
        try {
```

```
            GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
                    .name(databaseName)
                    .build();

            GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
            Instant createDate = response.database().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the database is " + createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getGlueTables(GlueClient glueClient, String dbName) {
        String myTableName = "";
        try {
            GetTablesRequest tableRequest = GetTablesRequest.builder()
                    .databaseName(dbName)
                    .build();

            GetTablesResponse response = glueClient.getTables(tableRequest);
            List<Table> tables = response.tableList();
            if (tables.isEmpty()) {
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return myTableName;
    }

    public static void startJob(GlueClient glueClient, String jobName, String
 inputDatabase, String inputTable,
            String outBucket) {
        try {
            Map<String, String> myMap = new HashMap<>();
            myMap.put("--input_database", inputDatabase);
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);

            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                    .workerType(WorkerType.G_1_X)
                    .numberOfWorkers(10)
                    .arguments(myMap)
                    .jobName(jobName)
                    .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is " +
 response.responseMetadata().requestId());

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createJob(GlueClient glueClient, String jobName, String iam,
 String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                    .pythonVersion("3")
                    .name("glueetl")
                    .scriptLocation(scriptLocation)
                    .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .description("A Job created by using the AWS SDK for Java V2")
                    .glueVersion("2.0")
                    .workerType(WorkerType.G_1_X)
                    .numberOfWorkers(10)
                    .name(jobName)
```

```
                    .role(iam)
                    .command(command)
                    .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAllJobs(GlueClient glueClient) {
        try {
            GetJobsRequest jobsRequest = GetJobsRequest.builder()
                    .maxResults(10)
                    .build();

            GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("Job name is : " + job.name());
                System.out.println("The job worker type is : " +
job.workerType().name());
            }

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                    .jobName(jobName)
                    .maxResults(20)
                    .build();

            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
```

```
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
 jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
 jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                    .jobName(jobName)
                    .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");
```

```
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteDatabase(GlueClient glueClient, String databaseName) {
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                    .name(databaseName)
                    .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
 DeleteCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [CreateCrawler]
  - [CreateJob]

- DeleteCrawler
- DeleteDatabase
- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

HealthImaging examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with HealthImaging.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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
- Scenarios

Actions

CopyImageSet

The following code example shows how to use CopyImageSet.

SDK for Java 2.x

```
    public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
            String datastoreId,
            String imageSetId,
            String latestVersionId,
            String destinationImageSetId,
            String destinationVersionId) {

        try {
            CopySourceImageSetInformation copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
                    .latestVersionId(latestVersionId)
                    .build();

            CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
                    .sourceImageSet(copySourceImageSetInformation);

            if (destinationImageSetId != null) {
                copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                        .imageSetId(destinationImageSetId)
                        .latestVersionId(destinationVersionId)
                        .build());
            }

            CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .sourceImageSetId(imageSetId)
                    .copyImageSetInformation(copyImageSetBuilder.build())
                    .build();

            CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

            return response.destinationImageSetProperties().imageSetId();
```

```
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- For API details, see [CopyImageSet](#) in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

`CreateDatastore`

The following code example shows how to use `CreateDatastore`.

SDK for Java 2.x

```
    public static String createMedicalImageDatastore(MedicalImagingClient
 medicalImagingClient,
            String datastoreName) {
        try {
            CreateDatastoreRequest datastoreRequest =
 CreateDatastoreRequest.builder()
                    .datastoreName(datastoreName)
                    .build();
            CreateDatastoreResponse response =
 medicalImagingClient.createDatastore(datastoreRequest);
            return response.datastoreId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- For API details, see CreateDatastore in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

DeleteDatastore

The following code example shows how to use `DeleteDatastore`.

SDK for Java 2.x

```
    public static void deleteMedicalImagingDatastore(MedicalImagingClient
 medicalImagingClient,
            String datastoreID) {
        try {
            DeleteDatastoreRequest datastoreRequest =
 DeleteDatastoreRequest.builder()
                    .datastoreId(datastoreID)
                    .build();
            medicalImagingClient.deleteDatastore(datastoreRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteDatastore in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

DeleteImageSet

The following code example shows how to use `DeleteImageSet`.

SDK for Java 2.x

```
    public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
            String datastoreId,
            String imagesetId) {
        try {
            DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId)
                    .build();

            medicalImagingClient.deleteImageSet(deleteImageSetRequest);

            System.out.println("The image set was deleted.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteImageSet in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

`GetDICOMImportJob`

The following code example shows how to use `GetDICOMImportJob`.

SDK for Java 2.x

```
    public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
medicalImagingClient,
            String datastoreId,
            String jobId) {
```

```
        try {
            GetDicomImportJobRequest getDicomImportJobRequest =
GetDicomImportJobRequest.builder()
                    .datastoreId(datastoreId)
                    .jobId(jobId)
                    .build();
            GetDicomImportJobResponse response =
medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
            return response.jobProperties();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- For API details, see GetDICOMImportJob in *AWS SDK for Java 2.x API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

`GetDatastore`

The following code example shows how to use `GetDatastore`.

SDK for Java 2.x

```
    public static DatastoreProperties getMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
            String datastoreID) {
        try {
            GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
                    .datastoreId(datastoreID)
                    .build();
            GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
            return response.datastoreProperties();
```

```
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- For API details, see GetDatastore in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

GetImageFrame

The following code example shows how to use `GetImageFrame`.

SDK for Java 2.x

```
        public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
                        String destinationPath,
                        String datastoreId,
                        String imagesetId,
                        String imageFrameId) {

                try {
                        GetImageFrameRequest getImageSetMetadataRequest =
GetImageFrameRequest.builder()
                                        .datastoreId(datastoreId)
                                        .imageSetId(imagesetId)

.imageFrameInformation(ImageFrameInformation.builder()
                                                        .imageFrameId(imageFrameId)
                                                        .build())
                                        .build();

medicalImagingClient.getImageFrame(getImageSetMetadataRequest,
```

```
FileSystems.getDefault().getPath(destinationPath));

                        System.out.println("Image frame downloaded to " +
destinationPath);
                } catch (MedicalImagingException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
```

- For API details, see GetImageFrame in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

GetImageSet

The following code example shows how to use `GetImageSet`.

SDK for Java 2.x

```
    public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient
medicalImagingClient,
            String datastoreId,
            String imagesetId,
            String versionId) {
        try {
            GetImageSetRequest.Builder getImageSetRequestBuilder =
GetImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId);

            if (versionId != null) {
                getImageSetRequestBuilder =
getImageSetRequestBuilder.versionId(versionId);
            }
```

```
            return
medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- For API details, see [GetImageSet](#) in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

`GetImageSetMetadata`

The following code example shows how to use `GetImageSetMetadata`.

SDK for Java 2.x

```
    public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
            String destinationPath,
            String datastoreId,
            String imagesetId,
            String versionId) {

        try {
            GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder =
GetImageSetMetadataRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId);

            if (versionId != null) {
                getImageSetMetadataRequestBuilder =
getImageSetMetadataRequestBuilder.versionId(versionId);
            }
```

```
medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
                   FileSystems.getDefault().getPath(destinationPath));

           System.out.println("Metadata downloaded to " + destinationPath);
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- For API details, see GetImageSetMetadata in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

`ListDICOMImportJobs`

The following code example shows how to use `ListDICOMImportJobs`.

SDK for Java 2.x

```
   public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
           String datastoreId) {

       try {
           ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
                   .datastoreId(datastoreId)
                   .build();
           ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
           return response.jobSummaries();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
```

```
        return new ArrayList<>();
    }
```

- For API details, see ListDICOMImportJobs in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

ListDatastores

The following code example shows how to use `ListDatastores`.

SDK for Java 2.x

```
    public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
        try {
            ListDatastoresRequest datastoreRequest = ListDatastoresRequest.builder()
                    .build();
            ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
            List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

            responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

            return datastoreSummaries;
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- For API details, see ListDatastores in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/medicalimaging#code-examples).

ListImageSetVersions

The following code example shows how to use `ListImageSetVersions`.

SDK for Java 2.x

```
    public static List<ImageSetProperties>
listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
            String datastoreId,
            String imagesetId) {
        try {
            ListImageSetVersionsRequest getImageSetRequest =
ListImageSetVersionsRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId)
                    .build();

            ListImageSetVersionsIterable responses = medicalImagingClient
                    .listImageSetVersionsPaginator(getImageSetRequest);
            List<ImageSetProperties> imageSetProperties = new ArrayList<>();
            responses.stream().forEach(response ->
imageSetProperties.addAll(response.imageSetPropertiesList()));

            return imageSetProperties;
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- For API details, see [ListImageSetVersions](https://docs.aws.amazon.com/goto/SdkForJavaV2/medical-imaging-2023-07-19/ListImageSetVersions) in *AWS SDK for Java 2.x API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/medicalimaging#code-examples).

ListTagsForResource

The following code example shows how to use `ListTagsForResource`.

SDK for Java 2.x

```
    public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .build();

            return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- For API details, see [ListTagsForResource](https://docs.aws.amazon.com/goto/SdkForJavaV2/medical-imaging-2023-07-19/ListTagsForResource) in *AWS SDK for Java 2.x API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/medicalimaging#code-examples).

`SearchImageSets`

The following code example shows how to use `SearchImageSets`.

SDK for Java 2.x

The utility function for searching image sets.

```
    public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
            MedicalImagingClient medicalImagingClient,
            String datastoreId, SearchCriteria searchCriteria) {
        try {
            SearchImageSetsRequest datastoreRequest =
 SearchImageSetsRequest.builder()
                    .datastoreId(datastoreId)
                    .searchCriteria(searchCriteria)
                    .build();
            SearchImageSetsIterable responses = medicalImagingClient
                    .searchImageSetsPaginator(datastoreRequest);
            List<ImageSetsMetadataSummary> imageSetsMetadataSummaries = new
 ArrayList<>();

            responses.stream().forEach(response -> imageSetsMetadataSummaries
                    .addAll(response.imageSetsMetadataSummaries()));

            return imageSetsMetadataSummaries;
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

Use case #1: EQUAL operator.

```
        List<SearchFilter> searchFilters =
 Collections.singletonList(SearchFilter.builder()
                .operator(Operator.EQUAL)
                .values(SearchByAttributeValue.builder()
                        .dicomPatientId(patientId)
                        .build())
```

```
                .build());

        SearchCriteria searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
searchMedicalImagingImageSets(
                medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets for patient " + patientId + " are:\n"
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Use case #2: BETWEEN operator using DICOMStudyDate and DICOMStudyTime.

```
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate("19990101")
                                        .dicomStudyTime("000000.000")
                                        .build())
                                .build(),
                        SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate((LocalDate.now()
                                                .format(formatter)))
                                        .dicomStudyTime("000000.000")
                                        .build())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();
```

```
        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println(
                    "The image sets searched with BETWEEN operator using
DICOMStudyDate and DICOMStudyTime are:\n"
                            +
                            imageSetsMetadataSummaries);
            System.out.println();
        }
```

Use case #3: BETWEEN operator using createdAt. Time studies were previously persisted.

```
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()
                                .createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))
                                .build(),
                        SearchByAttributeValue.builder()
                                .createdAt(Instant.now())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();
        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with BETWEEN operator using
createdAt are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Use case #4: EQUAL operator on DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response in ASC order on updatedAt field.

```
        Instant startDate = Instant.parse("1985-04-12T23:20:50.52Z");
```

```
        Instant endDate = Instant.now();

        searchFilters = Arrays.asList(
                SearchFilter.builder()
                        .operator(Operator.EQUAL)
                        .values(SearchByAttributeValue.builder()
                                .dicomSeriesInstanceUID(seriesInstanceUID)
                                .build())
                        .build(),
                SearchFilter.builder()
                        .operator(Operator.BETWEEN)
                        .values(

SearchByAttributeValue.builder().updatedAt(startDate).build(),

SearchByAttributeValue.builder().updatedAt(endDate).build()
                        ).build());

        Sort sort =
Sort.builder().sortOrder(SortOrder.ASC).sortField(SortField.UPDATED_AT).build();

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .sort(sort)
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with EQUAL operator on
DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response\n" +
                    "in ASC order on updatedAt field are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

- For API details, see SearchImageSets in *AWS SDK for Java 2.x API Reference*.

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/medicalimaging#code-examples).

StartDICOMImportJob

The following code example shows how to use `StartDICOMImportJob`.

SDK for Java 2.x

```
    public static String startDicomImportJob(MedicalImagingClient
 medicalImagingClient,
            String jobName,
            String datastoreId,
            String dataAccessRoleArn,
            String inputS3Uri,
            String outputS3Uri) {

        try {
            StartDicomImportJobRequest startDicomImportJobRequest =
 StartDicomImportJobRequest.builder()
                    .jobName(jobName)
                    .datastoreId(datastoreId)
                    .dataAccessRoleArn(dataAccessRoleArn)
                    .inputS3Uri(inputS3Uri)
                    .outputS3Uri(outputS3Uri)
                    .build();
            StartDicomImportJobResponse response =
 medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
            return response.jobId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- For API details, see [StartDICOMImportJob](https://docs.aws.amazon.com/goto/SdkForJavaV2/medical-imaging-2023-07-19/StartDICOMImportJob) in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

TagResource

The following code example shows how to use `TagResource`.

SDK for Java 2.x

```
    public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [TagResource](#) in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

UntagResource

The following code example shows how to use `UntagResource`.

SDK for Java 2.x

```
    public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
            UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see UntagResource in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

UpdateImageSetMetadata

The following code example shows how to use `UpdateImageSetMetadata`.

SDK for Java 2.x

```
    public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
                                                     String datastoreId,
                                                     String imagesetId,
                                                     String versionId,
                                                     MetadataUpdates
metadataUpdates) {
        try {
            UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
                    .builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId)
                    .latestVersionId(versionId)
                    .updateImageSetMetadataUpdates(metadataUpdates)
                    .build();

            UpdateImageSetMetadataResponse response =
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

            System.out.println("The image set metadata was updated" + response);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Use case #1: Insert or update an attribute.

```
            final String insertAttributes = """
                    {
                      "SchemaVersion": 1.1,
                      "Study": {
                        "DICOM": {
                          "StudyDescription": "CT CHEST"
                        }
                      }
                    }
                    """;
            MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
```

```
                    .dicomUpdates(DICOMUpdates.builder()
                            .updatableAttributes(SdkBytes.fromByteBuffer(
                                    ByteBuffer.wrap(insertAttributes
                                            .getBytes(StandardCharsets.UTF_8))))
                            .build())
                    .build();

            updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
 imagesetId,
                    versionid, metadataInsertUpdates);
```

Use case #2: Remove an attribute.

```
            final String removeAttributes = """
                    {
                      "SchemaVersion": 1.1,
                      "Study": {
                        "DICOM": {
                          "StudyDescription": "CT CHEST"
                        }
                      }
                    }
                    """;
            MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                    .dicomUpdates(DICOMUpdates.builder()
                            .removableAttributes(SdkBytes.fromByteBuffer(
                                    ByteBuffer.wrap(removeAttributes
                                            .getBytes(StandardCharsets.UTF_8))))
                            .build())
                    .build();

            updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
 imagesetId,
                    versionid, metadataRemoveUpdates);
```

Use case #3: Remove an instance.

```
            final String removeInstance = """
                    {
                      "SchemaVersion": 1.1,
                      "Study": {
```

```
                         "Series": {
                           "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                             "Instances": {
                               "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1":
 {}
                             }
                           }
                         }
                       }
                     }
                     """;
             MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                     .dicomUpdates(DICOMUpdates.builder()
                             .removableAttributes(SdkBytes.fromByteBuffer(
                                     ByteBuffer.wrap(removeInstance
                                             .getBytes(StandardCharsets.UTF_8))))
                             .build())
                     .build();

             updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
 imagesetId,
                     versionid, metadataRemoveUpdates);
```

- For API details, see UpdateImageSetMetadata in *AWS SDK for Java 2.x API Reference*.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Scenarios

Tagging a data store

The following code example shows how to tag a HealthImaging data store.

SDK for Java 2.x

To tag a data store.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                ImmutableMap.of("Deployment", "Development"));
```

The utility function for tagging a resource.

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

To list tags for a data store.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
                                medicalImagingClient,
                                datastoreArn);
                if (result != null) {
                        System.out.println("Tags for resource: " + result.tags());
                }
```

The utility function for listing a resource's tags.

```
    public static ListTagsForResourceResponse
 listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .build();

            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

To untag a data store.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                Collections.singletonList("Deployment"));
```

The utility function for untagging a resource.

```
    public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
```

```
            UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - ListTagsForResource
  - TagResource
  - UntagResource

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Tagging an image set

The following code example shows how to tag a HealthImaging image set.

SDK for Java 2.x

To tag an image set.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
```

```
                                    ImmutableMap.of("Deployment", "Development"));
```

The utility function for tagging a resource.

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

To list tags for an image set.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
                                medicalImagingClient,
                                imageSetArn);
                if (result != null) {
                        System.out.println("Tags for resource: " + result.tags());
                }
```

The utility function for listing a resource's tags.

```
    public static ListTagsForResourceResponse
 listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .build();

            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

To untag an image set.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                Collections.singletonList("Deployment"));
```

The utility function for untagging a resource.

```
    public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
            UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
```

```
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - ListTagsForResource
  - TagResource
  - UntagResource

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

IAM examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with IAM.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello IAM

The following code examples show how to get started using IAM.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/iam#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
```

```
}
```

- For API details, see ListPolicies in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

AttachRolePolicy

The following code example shows how to use `AttachRolePolicy`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
 ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
 iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

            // Ensure that the policy is not attached to this role
```

```
            String polArn = "";
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
 this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);

            System.out.println("Successfully attached policy " + policyArn +
                    " to role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- For API details, see AttachRolePolicy in *AWS SDK for Java 2.x API Reference*.

`CreateAccessKey`

The following code example shows how to use `CreateAccessKey`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <user>\s

                Where:
                   user - An AWS IAM user that you can obtain from the AWS
 Management Console.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String keyId = createIAMAccessKey(iam, user);
        System.out.println("The Key Id is " + keyId);
        iam.close();
    }

    public static String createIAMAccessKey(IamClient iam, String user) {
```

```
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey().accessKeyId();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see CreateAccessKey in *AWS SDK for Java 2.x API Reference*.

CreateAccountAlias

The following code example shows how to use `CreateAccountAlias`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to create (for example, myawsaccount).
\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IamClient iam, String alias) {
        try {
            CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.createAccountAlias(request);
            System.out.println("Successfully created account alias: " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for Java 2.x API Reference*.

CreatePolicy

The following code example shows how to use `CreatePolicy`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreatePolicy {

    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
```

```
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static void main(String[] args) {

        final String usage = """
                Usage:
                    CreatePolicy <policyName>\s

                Where:
                    policyName - A unique policy name.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMPolicy(iam, policyName);
        System.out.println("Successfully created a policy with this ARN value: " +
 result);
        iam.close();
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
```

```
                    .policyName(policyName)
                    .policyDocument(PolicyDocument)
                    .build();

            CreatePolicyResponse response = iam.createPolicy(request);

            // Wait until the policy is created.
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
 iamWaiter.waitUntilPolicyExists(polRequest);

 waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Java 2.x API Reference*.

CreateRole

The following code example shows how to use `CreateRole`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
```

```
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
*   This example requires a trust policy document. For more information, see:
*   https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-roles/
*
*
*  In addition, set up your development environment, including your credentials.
*
*  For information, see this documentation topic:
*
*  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = """
                Usage:
                    <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
                    fileLocation - The location of the JSON document that represents
 the trust policy.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String rolename = args[0];
        String fileLocation = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMRole(iam, rolename, fileLocation);
```

```
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMRole(IamClient iam, String rolename, String
 fileLocation) throws Exception {
        try {
            JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(jsonObject.toJSONString())
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " + response.role().arn());

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static Object readJsonSimpleDemo(String filename) throws Exception {
        FileReader reader = new FileReader(filename);
        JSONParser jsonParser = new JSONParser();
        return jsonParser.parse(reader);
    }
}
```

- For API details, see CreateRole in *AWS SDK for Java 2.x API Reference*.

CreateUser

The following code example shows how to use `CreateUser`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username>\s

                Where:
                    username - The name of the user to create.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String username = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMUser(iam, username);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            CreateUserResponse response = iam.createUser(request);

            // Wait until the user is created.
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
 iamWaiter.waitUntilUserExists(userRequest);
            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user().userName();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see [CreateUser](#) in *AWS SDK for Java 2.x API Reference*.

`DeleteAccessKey`

The following code example shows how to use `DeleteAccessKey`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/iam#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessKey>\s

                Where:
                    username - The name of the user.\s
                    accessKey - The access key ID for the secret access key you want
 to delete.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String username = args[0];
        String accessKey = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        deleteKey(iam, username, accessKey);
        iam.close();
    }

    public static void deleteKey(IamClient iam, String username, String accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteAccessKey in *AWS SDK for Java 2.x API Reference*.

DeleteAccountAlias

The following code example shows how to use `DeleteAccountAlias`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMAccountAlias(iam, alias);
        iam.close();
    }

    public static void deleteIAMAccountAlias(IamClient iam, String alias) {
        try {
            DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
                    .accountAlias(alias)
```

```
                    .build();

            iam.deleteAccountAlias(request);
            System.out.println("Successfully deleted account alias " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- For API details, see DeleteAccountAlias in *AWS SDK for Java 2.x API Reference*.

DeletePolicy

The following code example shows how to use `DeletePolicy`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeletePolicy {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <policyARN>\s

                Where:
                    policyARN - A policy ARN value to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(policyARN)
                    .build();

            iam.deletePolicy(request);
            System.out.println("Successfully deleted the policy");

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- For API details, see DeletePolicy in *AWS SDK for Java 2.x API Reference*.

DeleteUser

The following code example shows how to use `DeleteUser`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/iam#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
```

```
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [DeleteUser](#) in *AWS SDK for Java 2.x API Reference*.

DetachRolePolicy

The following code example shows how to use `DetachRolePolicy`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        detachPolicy(iam, roleName, policyArn);
        System.out.println("Done");
        iam.close();
    }

    public static void detachPolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
```

```
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                    " from role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DetachRolePolicy in *AWS SDK for Java 2.x API Reference*.

ListAccessKeys

The following code example shows how to use `ListAccessKeys`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user for which access keys are
 retrieved.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listKeys(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void listKeys(IamClient iam, String userName) {
        try {
            boolean done = false;
            String newMarker = null;

            while (!done) {
                ListAccessKeysResponse response;

                if (newMarker == null) {
                    ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                            .userName(userName)
                            .build();
```

```
                    response = iam.listAccessKeys(request);

                } else {
                    ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                            .userName(userName)
                            .marker(newMarker)
                            .build();

                    response = iam.listAccessKeys(request);
                }

                for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                    System.out.format("Retrieved access key %s",
 metadata.accessKeyId());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListAccessKeys in *AWS SDK for Java 2.x API Reference*.

`ListAccountAliases`

The following code example shows how to use `ListAccountAliases`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for Java 2.x API Reference*.

ListUsers

The following code example shows how to use `ListUsers`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
```

```
        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
                if (newMarker == null) {
                    ListUsersRequest request = ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = iam.listUsers(request);
                }

                for (User user : response.users()) {
                    System.out.format("\n Retrieved user %s", user.userName());
                    AttachedPermissionsBoundary permissionsBoundary =
 user.permissionsBoundary();
                    if (permissionsBoundary != null)
                        System.out.format("\n Permissions boundary details %s",

 permissionsBoundary.permissionsBoundaryTypeAsString());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

- For API details, see ListUsers in *AWS SDK for Java 2.x API Reference*.

UpdateAccessKey

The following code example shows how to use `UpdateAccessKey`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateAccessKey {

    private static StatusType statusType;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessId> <status>\s
```

```
                Where:
                    username - The name of the user whose key you want to update.\s
                    accessId - The access key ID of the secret access key you want
 to update.\s
                    status - The status you want to assign to the secret access key.
\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessId = args[1];
        String status = args[2];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateKey(iam, username, accessId, status);
        System.out.println("Done");
        iam.close();
    }

    public static void updateKey(IamClient iam, String username, String accessId,
 String status) {
        try {
            if (status.toLowerCase().equalsIgnoreCase("active")) {
                statusType = StatusType.ACTIVE;
            } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                statusType = StatusType.INACTIVE;
            } else {
                statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
            }

            UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
                    .accessKeyId(accessId)
                    .userName(username)
                    .status(statusType)
                    .build();

            iam.updateAccessKey(request);
```

```
            System.out.printf("Successfully updated the status of access key %s to"
 +
                    "status %s for user %s", accessId, status, username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see UpdateAccessKey in *AWS SDK for Java 2.x API Reference*.

UpdateUser

The following code example shows how to use `UpdateUser`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <curName> <newName>\s

                Where:
                    curName - The current user name.\s
                    newName - An updated user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String curName = args[0];
        String newName = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateIAMUser(iam, curName, newName);
        System.out.println("Done");
        iam.close();
    }

    public static void updateIAMUser(IamClient iam, String curName, String newName)
 {
        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                    .userName(curName)
                    .newUserName(newName)
                    .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s", newName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see UpdateUser in *AWS SDK for Java 2.x API Reference*.

Scenarios

Build and manage a resilient service

The following code example shows how to create a load-balanced web service that returns book, movie, and song recommendations. The example shows how the service responds to failures, and how to restructure the service for more resilience when failures occur.

- Use an Amazon EC2 Auto Scaling group to create Amazon Elastic Compute Cloud (Amazon EC2) instances based on a launch template and to keep the number of instances in a specified range.
- Handle and distribute HTTP requests with Elastic Load Balancing.
- Monitor the health of instances in an Auto Scaling group and forward requests only to healthy instances.
- Run a Python web server on each EC2 instance to handle HTTP requests. The web server responds with recommendations and health checks.
- Simulate a recommendation service with an Amazon DynamoDB table.
- Control web server response to requests and health checks by updating AWS Systems Manager parameters.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run the interactive scenario at a command prompt.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
```

```
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
 {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
 a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
```

```
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
                To keep things tidy and to avoid unwanted charges on your account,
 we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
 might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
 autoScaler, Database database)
```

```
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
 InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to create
 several AWS resources
                        to set up a load-balanced web service endpoint and explore
 some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
 provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances that
 each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
 across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
 targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
 tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

                The template also defines an IAM policy that each instance uses to
assume a role that grants
                permissions to access the DynamoDB recommendation table and Systems
Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
 The target group
                defines how the load balancer connects to instances. The load
 balancer provides a
                single endpoint where clients connect and dispatches requests to
 instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
 subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
 vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
 lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
 targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
 that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
 IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();
```

```
                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }
```

```
        System.out.println("Press Enter when you're ready to continue with the
 demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
 InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
 different parts of the system
                        to create situations where the web service fails, and shows
 how using a resilient
                        architecture can keep the web service running in spite of
 these failures.

                        At the start, the load balancer endpoint returns
 recommendations and reports that all targets are healthy.
                        """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
 recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
 named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
 set this parameter to name a non-existent table.
                        """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
```

```
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                       """);
       demoChoices(loadBalancer);

       System.out.println(
               """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
                       While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                       """);
       paramHelper.put(paramHelper.failureResponse, "static");

       System.out.println("""
               Now, sending a GET request to the load balancer endpoint returns a
static response.
               The service still reports as healthy because health checks are still
shallow.
               """);
       demoChoices(loadBalancer);

       System.out.println("Let's reinstate the recommendation service.");
       paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

       System.out.println("""
               Let's also substitute bad credentials for one of the instances in
the target group so that it can't
               access the DynamoDB recommendation table. We will get an instance id
value.
               """);

       LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
       AutoScaler autoScaler = new AutoScaler();

       // Create a new instance profile based on badCredsProfileName.
       templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
       String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
       System.out.println("The bad instance id values used for this demo is " +
badInstanceId);
```

```
        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
```

```
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
```

```
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
 one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
 loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
 HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
 loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
 InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
```

```
                    }
                    reader.close();

                    // Print the formatted JSON response.
                    System.out.println("Full Response:\n");
                    System.out.println(jsonResponse.toString());

                    // Close the HTTP client.
                    httpClient.close();

                }
                case 1 -> {
                    System.out.println("\nChecking the health of load balancer
targets:\n");
                    List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                    for (TargetHealthDescription target : health) {
                        System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
                    }
                    System.out.println("""
                            Note that it can take a minute or two for the health
check to update
                            after changes are made.
                            """);
                }
                case 2 -> {
                    System.out.println("\nOkay, let's move on.");
                    System.out.println("-".repeat(88));
                    return; // Exit the method when choice is 2
                }
                default -> System.out.println("You must choose a value between
0-2. Please select again.");
            }

        } catch (java.util.InputMismatchException e) {
            System.out.println("Invalid input. Please select again.");
            scanner.nextLine(); // Clear the input buffer.
        }
    }
}
```

```
    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Create a class that wraps Auto Scaling and Amazon EC2 actions.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }
```

```
    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
 TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


 getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
 newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
 iamInstanceProfile =
 software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
 is a valid IAM Instance Profile
```

```
                                                        // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
```

```
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
```

```
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
 getInstanceProfileRequest =
 software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
 getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
 rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
```

```
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
```

```
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
 DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
 securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
 cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }
```

```
                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
 autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
 zonesRequest =
 software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
 getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
 zonesResponse.availabilityZones().stream()

 .map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
 LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
 CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
```

```
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
 DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
 getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
 DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
```

```
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
```

```
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Create a class that wraps Elastic Load Balancing actions.

```
public class LoadBalancer {
```

```
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
```

```
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();
```

```
        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();
```

```
        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();
```

```
            System.out.println("Waiting for Load Balancer " + lbName + " to become
 available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
 loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .defaultActions(action)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Create a class that uses DynamoDB to simulate a recommendation service.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
```

```
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");
```

```
            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws IOException
 {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Create a class that wraps Systems Manager actions.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener

- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Create a user and assume a role

The following code example shows how to create a user and assume a role.

> **Warning**
>
> To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as AWS IAM Identity Center.

- Create a user with no permissions.

- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list S3 buckets using temporary credentials, then clean up resources.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create functions that wrap IAM user actions.

```
/*
  To run this Java V2 code example, set up your development environment, including
 your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3 Service
 client object with the temporary credentials.
  6. Deletes the resources.
 */

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
```

```
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = """

                Usage:
                    <username> <policyName> <roleName> <roleSessionName> 
<bucketName>\s

                Where:
                    username - The name of the IAM user to create.\s
                    policyName - The name of the policy to create.\s
                    roleName - The name of the role to create.\s
                    roleSessionName - The name of the session required for the 
assumeRole operation.\s
                    bucketName - The name of the Amazon S3 bucket from which objects 
are read.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        String policyName = args[1];
        String roleName = args[2];
        String roleSessionName = args[3];
        String bucketName = args[4];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
```

```
        System.out.println("Welcome to the AWS IAM example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 1. Create the IAM user.");
        User createUser = createIAMUser(iam, userName);

        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                " \"AWS\": \"" + userArn + "\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the role.");
        System.out.println("Perform an Amazon S3 Service operation using the
 temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();
```

```
            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static String createIAMRole(IamClient iam, String rolename, String json)
{

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(json)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " + response.role().arn());
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
```

```
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
 iamWaiter.waitUntilPolicyExists(polRequest);

 waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
 ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
 iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
            String polArn;
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
 this role.");
                    return;
                }
            }
```

```
            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);
            System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
            String keySecret) {

        // Use the creds of the new IAM user that was created in this code example.
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();

        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();

            // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
            // invoking assumeRole.
```

```
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()
                    .credentialsProvider(

StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                    .region(region)
                    .build();

            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
            }

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteRole(IamClient iam, String roleName, String polArn) {

        try {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                    .policyArn(polArn)
                    .roleName(roleName)
                    .build();

            iam.detachRolePolicy(rolePolicyRequest);

            // Delete the policy.
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(polArn)
                    .build();
```

```
            iam.deletePolicy(request);
            System.out.println("*** Successfully deleted " + polArn);

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteKey(IamClient iam, String username, String accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);
```

```
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Work with the IAM Policy Builder API

The following code example shows how to:

- Create IAM policies by using the object-oriented API.
- Use the IAM Policy Builder API with the IAM service.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

The examples use the following imports.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

Create a time-based policy.

```
        public String timeBasedPolicyExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")
                                                .addResource(IamResource.ALL)
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.DATE_GREATER_THAN)

 .key("aws:CurrentTime")

 .value("2020-04-01T00:00:00Z"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.DATE_LESS_THAN)
```

```
.key("aws:CurrentTime")

.value("2020-06-30T23:59:59Z")))
                                .build();

                // Use an IamPolicyWriter to write out the JSON string to a more
 readable
                // format.
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true)
                                .build());
        }
```

Create a policy with multiple conditions.

```
        public String multipleConditionsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")
                                                .addAction("dynamodb:BatchGetItem")
                                                .addAction("dynamodb:Query")
                                                .addAction("dynamodb:PutItem")
                                                .addAction("dynamodb:UpdateItem")
                                                .addAction("dynamodb:DeleteItem")

 .addAction("dynamodb:BatchWriteItem")

 .addResource("arn:aws:dynamodb:*:*:table/table-name")

 .addConditions(IamConditionOperator.STRING_EQUALS

 .addPrefix("ForAllValues:"),

 "dynamodb:Attributes",
                                                                List.of("column-
name1", "column-name2", "column-name3"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS
```

```
.addSuffix("IfExists"))

.key("dynamodb:Select")

.value("SPECIFIC_ATTRIBUTES")))
                                .build();

                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Use principals in a policy.

```
        public String specifyPrincipalsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.DENY)
                                                .addAction("s3:*")
                                                .addPrincipal(IamPrincipal.ALL)

.addResource("arn:aws:s3:::BUCKETNAME/*")

.addResource("arn:aws:s3:::BUCKETNAME")
                                                .addCondition(b1 -> b1

.operator(IamConditionOperator.ARN_NOT_EQUALS)

.key("aws:PrincipalArn")

.value("arn:aws:iam::444455556666:user/user-name")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Allow cross-account access.

```
        public String allowCrossAccountAccessExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
```

```
                                                            .effect(IamEffect.ALLOW)
                                                            .addPrincipal(IamPrincipalType.AWS,
 "111122223333")
                                                            .addAction("s3:PutObject")
                                                            .addResource("arn:aws:s3:::DOC-
EXAMPLE-BUCKET/*")
                                                            .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS)
                                                                            .key("s3:x-amz-acl")
                                                                            .value("bucket-
owner-full-control")))
                                            .build();
                return policy.toJson(IamPolicyWriter.builder()
                                            .prettyPrint(true).build());
        }
```

Build and upload an `IamPolicy`.

```
        public String createAndUploadPolicyExample(IamClient iam, String accountID,
 String policyName) {
                // Build the policy.
                IamPolicy policy = IamPolicy.builder() // 'version' defaults to
 "2012-10-17".
                                            .addStatement(IamStatement.builder()
                                                            .effect(IamEffect.ALLOW)
                                                            .addAction("dynamodb:PutItem")
                                                            .addResource("arn:aws:dynamodb:us-
east-1:" + accountID
                                                                            + ":table/
exampleTableName")
                                                            .build())
                                            .build();
                // Upload the policy.
                iam.createPolicy(r ->
 r.policyName(policyName).policyDocument(policy.toJson()));
                return
 policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

Download and work with an `IamPolicy`.

```
        public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName,
                        String newPolicyName) {

                String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
policyName;
                GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

                String policyVersion =
getPolicyResponse.policy().defaultVersionId();
                GetPolicyVersionResponse getPolicyVersionResponse = iam
                                .getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

                // Create an IamPolicy instance from the JSON string returned from
IAM.
                String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
                                StandardCharsets.UTF_8);
                IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

                /*
                 * All IamPolicy components are immutable, so use the copy method
that creates a
                 * new instance that
                 * can be altered in the same method call.
                 *
                 * Add the ability to get an item from DynamoDB as an additional
action.
                 */
                IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

                // Create a new statement that replaces the original statement.
                IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

                // Upload the new policy. IAM now has both policies.
                iam.createPolicy(r -> r.policyName(newPolicyName)
                                .policyDocument(newPolicy.toJson()));
```

```
                return
newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

- For more information, see AWS SDK for Java 2.x Developer Guide.
- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreatePolicy
  - GetPolicy
  - GetPolicyVersion

AWS IoT examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS IoT.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello AWS IoT

The following code examples show how to get started using AWS IoT.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
 Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings( IotClient iotClient) {
        ListThingsRequest thingsRequest = ListThingsRequest.builder()
            .maxResults(10)
            .build();

        ListThingsResponse response = iotClient.listThings(thingsRequest) ;
        List<ThingAttribute> thingList = response.things();
        for (ThingAttribute attribute : thingList) {
            System.out.println("Thing name: "+attribute.thingName());
            System.out.println("Thing ARN: "+attribute.thingArn());
        }
    }
}
```

- For API details, see listThings in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

AttachThingPrincipal

The following code example shows how to use `AttachThingPrincipal`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void attachCertificateToThing(IotClient iotClient, String
 thingName, String certificateArn) {
        // Attach the certificate to the thing.
        AttachThingPrincipalRequest principalRequest =
 AttachThingPrincipalRequest.builder()
            .thingName(thingName)
            .principal(certificateArn)
            .build();

        AttachThingPrincipalResponse attachResponse =
 iotClient.attachThingPrincipal(principalRequest);

        // Verify the attachment was successful.
        if (attachResponse.sdkHttpResponse().isSuccessful()) {
            System.out.println("Certificate attached to Thing successfully.");

            // Print additional information about the Thing.
            describeThing(iotClient, thingName);
        } else {
            System.err.println("Failed to attach certificate to Thing. HTTP Status
 Code: " +
                attachResponse.sdkHttpResponse().statusCode());
        }
    }
```

- For API details, see AttachThingPrincipal in *AWS SDK for Java 2.x API Reference*.

CreateKeysAndCertificate

The following code example shows how to use `CreateKeysAndCertificate`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/iot#code-examples).

```
    public static String createCertificate(IotClient iotClient) {
        try {
            CreateKeysAndCertificateResponse response =
 iotClient.createKeysAndCertificate();
            String certificatePem = response.certificatePem();
            String certificateArn = response.certificateArn();

            // Print the details.
            System.out.println("\nCertificate:");
            System.out.println(certificatePem);
            System.out.println("\nCertificate ARN:");
            System.out.println(certificateArn);
            return certificateArn;

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- For API details, see [CreateKeysAndCertificate] in *AWS SDK for Java 2.x API Reference*.

CreateThing

The following code example shows how to use `CreateThing`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/iot#code-examples).

```
    public static void createIoTThing(IotClient iotClient, String thingName) {
        try {
            CreateThingRequest createThingRequest = CreateThingRequest.builder()
                .thingName(thingName)
                .build();

            CreateThingResponse createThingResponse =
 iotClient.createThing(createThingRequest);
            System.out.println(thingName +" was successfully created. The ARN value
 is " + createThingResponse.thingArn());

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [CreateThing](https://docs.aws.amazon.com/goto/SdkForJavaV2/iot-2015-05-28/CreateThing) in *AWS SDK for Java 2.x API Reference*.

`CreateTopicRule`

The following code example shows how to use `CreateTopicRule`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/iot#code-examples).

```
    public static void createIoTRule(IotClient iotClient, String roleARN, String
 ruleName, String action) {
        try {
            String sql = "SELECT * FROM '" + TOPIC + "'";
            SnsAction action1 = SnsAction.builder()
                .targetArn(action)
                .roleArn(roleARN)
                .build();

            // Create the action.
            Action myAction = Action.builder()
                .sns(action1)
                .build();

            // Create the topic rule payload.
            TopicRulePayload topicRulePayload = TopicRulePayload.builder()
                .sql(sql)
                .actions(myAction)
                .build();

            // Create the topic rule request.
            CreateTopicRuleRequest topicRuleRequest =
 CreateTopicRuleRequest.builder()
                .ruleName(ruleName)
                .topicRulePayload(topicRulePayload)
                .build();

            // Create the rule.
            iotClient.createTopicRule(topicRuleRequest);
            System.out.println("IoT Rule created successfully.");

        } catch (IotException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see CreateTopicRule in *AWS SDK for Java 2.x API Reference*.

DeleteCertificate

The following code example shows how to use `DeleteCertificate`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/iot#code-examples).

```
    public static void deleteCertificate(IotClient iotClient, String
 certificateArn ) {
        DeleteCertificateRequest certificateProviderRequest =
 DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        iotClient.deleteCertificate(certificateProviderRequest);
        System.out.println(certificateArn +" was successfully deleted.");
    }
```

- For API details, see DeleteCertificate in *AWS SDK for Java 2.x API Reference*.

DeleteThing

The following code example shows how to use `DeleteThing`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteIoTThing(IotClient iotClient, String thingName) {
```

```
        try {
            DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
                .thingName(thingName)
                .build();

            iotClient.deleteThing(deleteThingRequest);
            System.out.println("Deleted Thing " + thingName);

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteThing in *AWS SDK for Java 2.x API Reference*.

`DescribeEndpoint`

The following code example shows how to use `DescribeEndpoint`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String describeEndpoint(IotClient iotClient) {
        try {
            DescribeEndpointResponse endpointResponse =
 iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

            // Get the endpoint URL.
            String endpointUrl = endpointResponse.endpointAddress();
            String exString = getValue(endpointUrl);
            String fullEndpoint = "https://"+exString+"-ats.iot.us-
east-1.amazonaws.com";

            System.out.println("Full Endpoint URL: "+fullEndpoint);
```

```
            return fullEndpoint;

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "" ;
    }
```

- For API details, see DescribeEndpoint in *AWS SDK for Java 2.x API Reference*.

`DescribeThing`

The following code example shows how to use `DescribeThing`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    private static void describeThing(IotClient iotClient, String thingName) {
        try {
            DescribeThingRequest thingRequest = DescribeThingRequest.builder()
                .thingName(thingName)
                .build() ;

            // Print Thing details.
            DescribeThingResponse describeResponse =
 iotClient.describeThing(thingRequest);
            System.out.println("Thing Details:");
            System.out.println("Thing Name: " + describeResponse.thingName());
            System.out.println("Thing ARN: " + describeResponse.thingArn());

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeThing in *AWS SDK for Java 2.x API Reference*.

DetachThingPrincipal

The following code example shows how to use `DetachThingPrincipal`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
        try {
            DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
                .principal(certificateArn)
                .thingName(thingName)
                .build();

            iotClient.detachThingPrincipal(thingPrincipalRequest);
            System.out.println(certificateArn +" was successfully removed from "
+thingName);

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DetachThingPrincipal in *AWS SDK for Java 2.x API Reference*.

ListCertificates

The following code example shows how to use `ListCertificates`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}
```

- For API details, see ListCertificates in *AWS SDK for Java 2.x API Reference*.

`SearchIndex`

The following code example shows how to use `SearchIndex`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();

    try {
        // Perform the search and get the result.
```

```
            SearchIndexResponse searchIndexResponse =
iotClient.searchIndex(searchIndexRequest);

            // Process the result.
            if (searchIndexResponse.things().isEmpty()) {
                System.out.println("No things found.");
            } else {
                searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
            }
        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see SearchIndex in *AWS SDK for Java 2.x API Reference*.

`UpdateThing`

The following code example shows how to use `UpdateThing`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void updateThing(IotClient iotClient, String thingName) {
        // Specify the new attribute values.
        String newLocation = "Office";
        String newFirmwareVersion = "v2.0";

        Map<String, String> attMap = new HashMap<>();
        attMap.put("location", newLocation);
        attMap.put("firmwareVersion", newFirmwareVersion);

        AttributePayload attributePayload = AttributePayload.builder()
            .attributes(attMap)
```

```
                .build();

        UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
                .thingName(thingName)
                .attributePayload(attributePayload)
                .build();

        try {
            // Update the IoT Thing attributes.
            iotClient.updateThing(updateThingRequest);
            System.out.println("Thing attributes updated successfully.");

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see UpdateThing in *AWS SDK for Java 2.x API Reference*.

Scenarios

Work with device management use cases

The following code example shows how to work with AWS IoT device management use cases using AWS IoT SDK

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
```

```
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.AttributePayload;
import software.amazon.awssdk.services.iot.model.Certificate;
import software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iot.model.UpdateThingRequest;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import java.net.URI;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 *
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach amd delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String TOPIC = "your-iot-topic";
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <roleARN> <snsAction>

                Where:
                    roleARN - The ARN of an IAM role that has permission to work
 with AWS IOT.
                    snsAction  - An ARN of an SNS topic.
                """;

       if (args.length != 2) {
           System.out.println(usage);
           System.exit(1);
       }

        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IoT example workflow.");
        System.out.println("""
            This example program demonstrates various interactions with the AWS 
Internet of Things (IoT) Core service. The program guides you through a series of 
steps,
            including creating an IoT Thing, generating a device certificate, 
updating the Thing with attributes, and so on.
            It utilizes the AWS SDK for Java V2 and incorporates functionality for 
creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS IoT 
capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

            """);
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an AWS IoT Thing.");
        System.out.println("""
            An AWS IoT Thing represents a virtual entity in the AWS IoT service that 
can be associated with a physical device.
            """);
        // Prompt the user for input.
        System.out.print("Enter Thing name: ");
        thingName = scanner.nextLine();
        createIoTThing(iotClient, thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Generate a device certificate.");
        System.out.println("""
            A device certificate performs a role in securing the communication 
between devices (Things) and the AWS IoT platform.
            """);

        System.out.print("Do you want to create a certificate for " +thingName +"? 
(y/n)");
        String certAns = scanner.nextLine();
        String certificateArn="" ;
        if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
```

```
            certificateArn = createCertificate(iotClient);
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            attachCertificateToThing(iotClient, thingName, certificateArn);
        } else {
            System.out.println("A device certificate was not created.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Update an AWS IoT Thing with Attributes.");
        System.out.println("""
             IoT Thing attributes, represented as key-value pairs, offer a pivotal
 advantage in facilitating efficient data
             management and retrieval within the AWS IoT ecosystem.
            """);
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        updateThing(iotClient, thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Return a unique endpoint specific to the Amazon Web
 Services account.");
        System.out.println("""
            An IoT Endpoint refers to a specific URL or Uniform Resource Locator
 that serves as the entry point for communication between IoT devices and the AWS
 IoT service.
           """);
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        String endpointUrl = describeEndpoint(iotClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        if (certificateArn.length() > 0) {
            listCertificates(iotClient);
        } else {
            System.out.println("You did not create a certificates. Skipping this
 step.");
        }
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
        System.out.println("""
            A Thing Shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
            of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
            the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a Thing Shadow.
           """);
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        IotDataPlaneClient iotPlaneClient = IotDataPlaneClient.builder()
            .region(Region.US_EAST_1)
            .endpointOverride(URI.create(endpointUrl))
            .build();

        updateShadowThing(iotPlaneClient, thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON format.");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        getPayload(iotPlaneClient, thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Creates a rule");
        System.out.println("""
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """);
        System.out.print("Enter Rule name: ");
        ruleName = scanner.nextLine();
        createIoTRule(iotClient, roleARN, ruleName, snsAction);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        System.out.print("Press Enter to continue...");
```

```
        scanner.nextLine();
        listIoTRules(iotClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        String queryString = "thingName:"+thingName ;
        searchThings(iotClient, queryString);
        System.out.println(DASHES);

        System.out.println(DASHES);
        if (certificateArn.length() > 0) {
            System.out.print("Do you want to detach and delete the certificate for "
 +thingName +"? (y/n)");
            String delAns = scanner.nextLine();
            if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
                System.out.println("11. You selected to detach amd delete the
 certificate.");
                System.out.print("Press Enter to continue...");
                scanner.nextLine();
                detachThingPrincipal(iotClient, thingName, certificateArn);
                deleteCertificate(iotClient, certificateArn);
            } else {
                System.out.println("11. You selected not to delete the
 certificate.");
            }
        } else {
            System.out.println("11. You did not create a certificate so there is
 nothing to delete.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete the AWS IoT Thing.");
        System.out.print("Do you want to delete the IoT Thing? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            deleteIoTThing(iotClient, thingName);
        } else {
            System.out.println("The IoT Thing was not deleted.");
        }
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("The AWS IoT workflow has successfully completed.");
        System.out.println(DASHES);
    }

    public static void listCertificates(IotClient iotClient) {
        ListCertificatesResponse response = iotClient.listCertificates();
        List<Certificate> certList = response.certificates();
        for (Certificate cert : certList) {
            System.out.println("Cert id: " + cert.certificateId());
            System.out.println("Cert Arn: " + cert.certificateArn());
        }
    }

    public static void listIoTRules(IotClient iotClient) {
        try {
            ListTopicRulesRequest listTopicRulesRequest =
 ListTopicRulesRequest.builder().build();
            ListTopicRulesResponse listTopicRulesResponse =
 iotClient.listTopicRules(listTopicRulesRequest);
            System.out.println("List of IoT Rules:");
            List<TopicRuleListItem> ruleList = listTopicRulesResponse.rules();
            for (TopicRuleListItem rule : ruleList) {
                System.out.println("Rule Name: " + rule.ruleName());
                System.out.println("Rule ARN: " + rule.ruleArn());
                System.out.println("--------------");
            }

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createIoTRule(IotClient iotClient, String roleARN, String
 ruleName, String action) {
        try {
            String sql = "SELECT * FROM '" + TOPIC + "'";
            SnsAction action1 = SnsAction.builder()
                .targetArn(action)
                .roleArn(roleARN)
                .build();
```

```
            // Create the action.
            Action myAction = Action.builder()
                .sns(action1)
                .build();

            // Create the topic rule payload.
            TopicRulePayload topicRulePayload = TopicRulePayload.builder()
                .sql(sql)
                .actions(myAction)
                .build();

            // Create the topic rule request.
            CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
                .ruleName(ruleName)
                .topicRulePayload(topicRulePayload)
                .build();

            // Create the rule.
            iotClient.createTopicRule(topicRuleRequest);
            System.out.println("IoT Rule created successfully.");

        } catch (IotException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
        try {
            GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
                .thingName(thingName)
                .build();

            GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

            // Extracting payload from response.
            SdkBytes payload = getThingShadowResponse.payload();
            String payloadString = payload.asUtf8String();
            System.out.println("Received Shadow Data: " + payloadString);
```

```
        } catch (IotException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
 thingName) {
        try {
            // Create Thing Shadow State Document.
            String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
 \"humidity\":50}}}";
            SdkBytes data= SdkBytes.fromString(stateDocument,
 StandardCharsets.UTF_8 );
            UpdateThingShadowRequest updateThingShadowRequest =
 UpdateThingShadowRequest.builder()
                .thingName(thingName)
                .payload(data)
                .build();

            // Update Thing Shadow.
            iotPlaneClient.updateThingShadow(updateThingShadowRequest);
            System.out.println("Thing Shadow updated successfully.");

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateThing(IotClient iotClient, String thingName) {
        // Specify the new attribute values.
        String newLocation = "Office";
        String newFirmwareVersion = "v2.0";

        Map<String, String> attMap = new HashMap<>();
        attMap.put("location", newLocation);
        attMap.put("firmwareVersion", newFirmwareVersion);

        AttributePayload attributePayload = AttributePayload.builder()
            .attributes(attMap)
            .build();

        UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
```

```
            .thingName(thingName)
            .attributePayload(attributePayload)
            .build();

        try {
            // Update the IoT Thing attributes.
            iotClient.updateThing(updateThingRequest);
            System.out.println("Thing attributes updated successfully.");

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String describeEndpoint(IotClient iotClient) {
        try {
            DescribeEndpointResponse endpointResponse =
 iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

            // Get the endpoint URL.
            String endpointUrl = endpointResponse.endpointAddress();
            String exString = getValue(endpointUrl);
            String fullEndpoint = "https://"+exString+"-ats.iot.us-
east-1.amazonaws.com";

            System.out.println("Full Endpoint URL: "+fullEndpoint);
            return fullEndpoint;

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "" ;
    }

    public static void detachThingPrincipal(IotClient iotClient, String thingName,
 String certificateArn){
        try {
            DetachThingPrincipalRequest thingPrincipalRequest =
 DetachThingPrincipalRequest.builder()
                .principal(certificateArn)
                .thingName(thingName)
                .build();
```

```
            iotClient.detachThingPrincipal(thingPrincipalRequest);
            System.out.println(certificateArn +" was successfully removed from "
+thingName);

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        iotClient.deleteCertificate(certificateProviderRequest);
        System.out.println(certificateArn +" was successfully deleted.");
    }

    // Get the cert Id  from the Cert ARN value.
    private static String extractCertificateId(String certificateArn) {
        // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
        String[] arnParts = certificateArn.split(":");
        String certificateIdPart = arnParts[arnParts.length - 1];
        return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1);
    }

    public static String createCertificate(IotClient iotClient) {
        try {
            CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
            String certificatePem = response.certificatePem();
            String certificateArn = response.certificateArn();

            // Print the details.
            System.out.println("\nCertificate:");
            System.out.println(certificatePem);
            System.out.println("\nCertificate ARN:");
            System.out.println(certificateArn);
            return certificateArn;
```

```
        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }

    public static void attachCertificateToThing(IotClient iotClient, String
 thingName, String certificateArn) {
        // Attach the certificate to the thing.
        AttachThingPrincipalRequest principalRequest =
 AttachThingPrincipalRequest.builder()
            .thingName(thingName)
            .principal(certificateArn)
            .build();

        AttachThingPrincipalResponse attachResponse =
 iotClient.attachThingPrincipal(principalRequest);

        // Verify the attachment was successful.
        if (attachResponse.sdkHttpResponse().isSuccessful()) {
            System.out.println("Certificate attached to Thing successfully.");

            // Print additional information about the Thing.
            describeThing(iotClient, thingName);
        } else {
            System.err.println("Failed to attach certificate to Thing. HTTP Status
 Code: " +
                attachResponse.sdkHttpResponse().statusCode());
        }
    }

    private static void describeThing(IotClient iotClient, String thingName) {
        try {
            DescribeThingRequest thingRequest = DescribeThingRequest.builder()
                .thingName(thingName)
                .build() ;

            // Print Thing details.
            DescribeThingResponse describeResponse =
 iotClient.describeThing(thingRequest);
            System.out.println("Thing Details:");
            System.out.println("Thing Name: " + describeResponse.thingName());
```

```
            System.out.println("Thing ARN: " + describeResponse.thingArn());

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteIoTThing(IotClient iotClient, String thingName) {
        try {
            DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
                .thingName(thingName)
                .build();

            iotClient.deleteThing(deleteThingRequest);
            System.out.println("Deleted Thing " + thingName);

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createIoTThing(IotClient iotClient, String thingName) {
        try {
            CreateThingRequest createThingRequest = CreateThingRequest.builder()
                .thingName(thingName)
                .build();

            CreateThingResponse createThingResponse =
 iotClient.createThing(createThingRequest);
            System.out.println(thingName +" was successfully created. The ARN value
 is " + createThingResponse.thingArn());

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    private static String getValue(String input) {
        // Define a regular expression pattern for extracting the subdomain.
        Pattern pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\
\.com");
```

```
        // Match the pattern against the input string.
        Matcher matcher = pattern.matcher(input);

        // Check if a match is found.
        if (matcher.find()) {
            // Extract the subdomain from the first capturing group.
            String subdomain = matcher.group(1);
            System.out.println("Extracted subdomain: " + subdomain);
            return subdomain ;
        } else {
            System.out.println("No match found");
        }
        return "" ;
    }

    public static void searchThings(IotClient iotClient, String queryString){
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        try {
            // Perform the search and get the result.
            SearchIndexResponse searchIndexResponse =
 iotClient.searchIndex(searchIndexRequest);

            // Process the result.
            if (searchIndexResponse.things().isEmpty()) {
                System.out.println("No things found.");
            } else {
                searchIndexResponse.things().forEach(thing ->
 System.out.println("Thing id found using search is " + thing.thingId()));
            }
        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

AWS IoT data examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS IoT data.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

`GetThingShadow`

The following code example shows how to use `GetThingShadow`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
        try {
            GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
                .thingName(thingName)
                .build();
```

```
            GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

            // Extracting payload from response.
            SdkBytes payload = getThingShadowResponse.payload();
            String payloadString = payload.asUtf8String();
            System.out.println("Received Shadow Data: " + payloadString);

        } catch (IotException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see GetThingShadow in *AWS SDK for Java 2.x API Reference*.

UpdateThingShadow

The following code example shows how to use `UpdateThingShadow`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
        try {
            // Create Thing Shadow State Document.
            String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
            SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
            UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
                .thingName(thingName)
                .payload(data)
                .build();
```

```
            // Update Thing Shadow.
            iotPlaneClient.updateThingShadow(updateThingShadowRequest);
            System.out.println("Thing Shadow updated successfully.");

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see UpdateThingShadow in *AWS SDK for Java 2.x API Reference*.

Amazon Keyspaces examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Keyspaces.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon Keyspaces

The following code examples show how to get started using Amazon Keyspaces.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKeyspaces {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        listKeyspaces(keyClient);
    }

    public static void listKeyspaces(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesResponse response =
 keyClient.listKeyspaces(keyspacesRequest);
            List<KeyspaceSummary> keyspaces = response.keyspaces();
            for (KeyspaceSummary keyspace : keyspaces) {
                System.out.println("The name of the keyspace is " +
 keyspace.keyspaceName());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- For API details, see ListKeyspaces in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

CreateKeyspace

The following code example shows how to use `CreateKeyspace`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
            System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
```

- For API details, see [CreateKeyspace](#) in *AWS SDK for Java 2.x API Reference*.

`CreateTable`

The following code example shows how to use `CreateTable`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void createTable(KeyspacesClient keyClient, String keySpace,
 String tableName) {
        try {
            // Set the columns.
            ColumnDefinition defTitle = ColumnDefinition.builder()
                    .name("title")
                    .type("text")
                    .build();

            ColumnDefinition defYear = ColumnDefinition.builder()
                    .name("year")
                    .type("int")
                    .build();

            ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                    .name("release_date")
                    .type("timestamp")
                    .build();

            ColumnDefinition defPlot = ColumnDefinition.builder()
                    .name("plot")
                    .type("text")
                    .build();
```

```
            List<ColumnDefinition> colList = new ArrayList<>();
            colList.add(defTitle);
            colList.add(defYear);
            colList.add(defReleaseDate);
            colList.add(defPlot);

            // Set the keys.
            PartitionKey yearKey = PartitionKey.builder()
                    .name("year")
                    .build();

            PartitionKey titleKey = PartitionKey.builder()
                    .name("title")
                    .build();

            List<PartitionKey> keyList = new ArrayList<>();
            keyList.add(yearKey);
            keyList.add(titleKey);

            SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                    .partitionKeys(keyList)
                    .allColumns(colList)
                    .build();

            PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                    .status(PointInTimeRecoveryStatus.ENABLED)
                    .build();

            CreateTableRequest tableRequest = CreateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .schemaDefinition(schemaDefinition)
                    .pointInTimeRecovery(timeRecovery)
                    .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is " + response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [CreateTable](#) in *AWS SDK for Java 2.x API Reference*.

DeleteKeyspace

The following code example shows how to use `DeleteKeyspace`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void deleteKeyspace(KeyspacesClient keyClient, String
 keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
 DeleteKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            keyClient.deleteKeyspace(deleteKeyspaceRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DeleteKeyspace](#) in *AWS SDK for Java 2.x API Reference*.

DeleteTable

The following code example shows how to use `DeleteTable`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
 String tableName) {
        try {
            DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            keyClient.deleteTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DeleteTable](#) in *AWS SDK for Java 2.x API Reference*.

`GetKeyspace`

The following code example shows how to use `GetKeyspace`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
 keyspaceName) {
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The " + name + " KeySpace is ready");

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [GetKeyspace](#) in *AWS SDK for Java 2.x API Reference*.

GetTable

The following code example shows how to use `GetTable`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
 String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
```

```
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see GetTable in *AWS SDK for Java 2.x API Reference*.

ListKeyspaces

The following code example shows how to use `ListKeyspaces`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
```

```
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesIterable listRes =
 keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                    .flatMap(r -> r.keyspaces().stream())
                    .forEach(content -> System.out.println(" Name: " +
 content.keyspaceName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListKeyspaces in *AWS SDK for Java 2.x API Reference*.

ListTables

The following code example shows how to use `ListTables`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            ListTablesIterable listRes =
 keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
```

```
                    .flatMap(r -> r.tables().stream())
                    .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                            " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListTables in *AWS SDK for Java 2.x API Reference*.

RestoreTable

The following code example shows how to use `RestoreTable`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
ZonedDateTime utc) {
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
                    .restoreTimestamp(myTime)
                    .sourceTableName("Movie")
                    .targetKeyspaceName(keyspaceName)
                    .targetTableName("MovieRestore")
                    .sourceKeyspaceName(keyspaceName)
                    .build();

            RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
response.restoredTableARN());
```

```
        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see RestoreTable in *AWS SDK for Java 2.x API Reference*.

UpdateTable

The following code example shows how to use `UpdateTable`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void updateTable(KeyspacesClient keyClient, String keySpace,
 String tableName) {
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                    .name("watched")
                    .type("boolean")
                    .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .addColumns(def)
                    .build();

            keyClient.updateTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
```

- For API details, see UpdateTable in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with keyspaces and tables

The following code example shows how to:

- Create a keyspace and table. The table schema holds movie data and has point-in-time recovery enabled.
- Connect to the keyspace using a secure TLS connection with SigV4 authentication.
- Query the table. Add, retrieve, and update movie data.
- Update the table. Add a column to track watched movies.
- Restore the table to its previous state and clean up resources.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Before running this Java code example, you must create a
 * Java keystore (JKS) file and place it in your project's resources folder.
 *
 * This file is a secure file format used to hold certificate information for
 * Java applications. This is required to make a connection to Amazon Keyspaces.
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Create a keyspace.
 * 2. Check for keyspace existence.
 * 3. List keyspaces using a paginator.
 * 4. Create a table with a simple movie data schema and enable point-in-time
 * recovery.
 * 5. Check for the table to be in an Active state.
 * 6. List all tables in the keyspace.
 * 7. Use a Cassandra driver to insert some records into the Movie table.
 * 8. Get all records from the Movie table.
 * 9. Get a specific Movie.
 * 10. Get a UTC timestamp for the current time.
 * 11. Update the table schema to add a ‘watched’ Boolean column.
 * 12. Update an item as watched.
 * 13. Query for items with watched = True.
 * 14. Restore the table back to the previous state using the timestamp.
 * 15. Check for completion of the restore action.
 * 16. Delete the table.
 * 17. Confirm that both tables are deleted.
 * 18. Delete the keyspace.
 */

public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    /*
     * Usage:
     * fileName - The name of the JSON file that contains movie data. (Get this file
     * from the GitHub repo at resources/sample_file.)
     * keyspaceName - The name of the keyspace to create.
     */
    public static void main(String[] args) throws InterruptedException, IOException
 {
        String fileName = "<Replace with the JSON file that contains movie data>";
        String keyspaceName = "<Replace with the name of the keyspace to create>";
        String titleUpdate = "The Family";
        int yearUpdate = 2013;
        String tableName = "Movie";
        String tableNameRestore = "MovieRestore";
```

```
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        DriverConfigLoader loader =
 DriverConfigLoader.fromClasspath("application.conf");
        CqlSession session = CqlSession.builder()
                .withConfigLoader(loader)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Keyspaces example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a keyspace.");
        createKeySpace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        Thread.sleep(5000);
        System.out.println("2. Check for keyspace existence.");
        checkKeyspaceExistence(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List keyspaces using a paginator.");
        listKeyspacesPaginator(keyClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Create a table with a simple movie data schema and
 enable point-in-time recovery.");
        createTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Check for the table to be in an Active state.");
        Thread.sleep(6000);
        checkTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("6. List all tables in the keyspace.");
        listTables(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Use a Cassandra driver to insert some records into
 the Movie table.");
        Thread.sleep(6000);
        loadData(session, fileName, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get all records from the Movie table.");
        getMovieData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get a specific Movie.");
        getSpecificMovie(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a UTC timestamp for the current time.");
        ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
        System.out.println("DATETIME = " + Date.from(utc.toInstant()));
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Update the table schema to add a watched Boolean
 column.");
        updateTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Update an item as watched.");
        Thread.sleep(10000); // Wait 10 secs for the update.
        updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Query for items with watched = True.");
        getWatchedData(session, keyspaceName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state using
 the timestamp.");
        System.out.println("Note that the restore operation can take up to 20
 minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String
 keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
 DeleteKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();
```

```
            keyClient.deleteKeyspace(deleteKeyspaceRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTableDelete(KeyspacesClient keyClient, String
 keyspaceName, String tableName)
            throws InterruptedException {
        try {
            String status;
            GetTableResponse response;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            // Keep looping until table cannot be found and a
 ResourceNotFoundException is
            // thrown.
            while (true) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);
                Thread.sleep(500);
            }

        } catch (ResourceNotFoundException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println("The table is deleted");
    }

    public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
 String tableName) {
        try {
            DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            keyClient.deleteTable(tableRequest);
```

```
        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkRestoredTable(KeyspacesClient keyClient, String
 keyspaceName, String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println("The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
 ZonedDateTime utc) {
```

```
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
                    .restoreTimestamp(myTime)
                    .sourceTableName("Movie")
                    .targetKeyspaceName(keyspaceName)
                    .targetTableName("MovieRestore")
                    .sourceKeyspaceName(keyspaceName)
                    .build();

            RestoreTableResponse response =
 keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
 response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getWatchedData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session
                .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
 watched = true ALLOW FILTERING;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    public static void updateRecord(CqlSession session, String keySpace, String
 titleUpdate, int yearUpdate) {
        String sqlStatement = "UPDATE \"" + keySpace
                + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year = :k1;";
        BatchStatementBuilder builder =
 BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
                .setString("k0", titleUpdate)
                .setInt("k1", yearUpdate)
                .build());
```

```
        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
    }

    public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                    .name("watched")
                    .type("boolean")
                    .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .addColumns(def)
                    .build();

            keyClient.updateTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificMovie(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute(
                "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title = 'The
Family' ALLOW FILTERING ;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Get records from the Movie table.
    public static void getMovieData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
"\".\"Movie\";");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
```

```
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Load data into the table.
    public static void loadData(CqlSession session, String fileName, String
 keySpace) throws IOException {
        String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
 year, plot) values (:k0, :k1, :k2)";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {

            // Add 20 movies to the table.
            if (t == 20)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String plot = currentNode.path("info").path("plot").toString();

            // Insert the data into the Amazon Keyspaces table.
            BatchStatementBuilder builder =
 BatchStatement.builder(DefaultBatchType.UNLOGGED);
            builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
            PreparedStatement preparedStatement = session.prepare(sqlStatement);
            builder.addStatement(preparedStatement.boundStatementBuilder()
                    .setString("k0", title)
                    .setInt("k1", year)
                    .setString("k2", plot)
                    .build());

            BatchStatement batchStatement = builder.build();
            session.execute(batchStatement);
            t++;
        }

        System.out.println("You have added " + t + " records successfully!");
```

```
    }

    public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                    .flatMap(r -> r.tables().stream())
                    .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                            " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }
```

```
            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createTable(KeyspacesClient keyClient, String keySpace,
 String tableName) {
        try {
            // Set the columns.
            ColumnDefinition defTitle = ColumnDefinition.builder()
                    .name("title")
                    .type("text")
                    .build();

            ColumnDefinition defYear = ColumnDefinition.builder()
                    .name("year")
                    .type("int")
                    .build();

            ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                    .name("release_date")
                    .type("timestamp")
                    .build();

            ColumnDefinition defPlot = ColumnDefinition.builder()
                    .name("plot")
                    .type("text")
                    .build();

            List<ColumnDefinition> colList = new ArrayList<>();
            colList.add(defTitle);
            colList.add(defYear);
            colList.add(defReleaseDate);
            colList.add(defPlot);

            // Set the keys.
```

```
            PartitionKey yearKey = PartitionKey.builder()
                    .name("year")
                    .build();

            PartitionKey titleKey = PartitionKey.builder()
                    .name("title")
                    .build();

            List<PartitionKey> keyList = new ArrayList<>();
            keyList.add(yearKey);
            keyList.add(titleKey);

            SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                    .partitionKeys(keyList)
                    .allColumns(colList)
                    .build();

            PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                    .status(PointInTimeRecoveryStatus.ENABLED)
                    .build();

            CreateTableRequest tableRequest = CreateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .schemaDefinition(schemaDefinition)
                    .pointInTimeRecovery(timeRecovery)
                    .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is " + response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();
```

```
            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                    .flatMap(r -> r.keyspaces().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The " + name + " KeySpace is ready");

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
            System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
                System.exit(1);
            }
        }
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Kinesis examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Kinesis.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

CreateStream

The following code example shows how to use `CreateStream`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.CreateStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataStream {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream (for example,
 StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();
        createStream(kinesisClient, streamName);
        System.out.println("Done");
        kinesisClient.close();
    }

    public static void createStream(KinesisClient kinesisClient, String streamName)
 {
        try {
            CreateStreamRequest streamReq = CreateStreamRequest.builder()
                    .streamName(streamName)
                    .shardCount(1)
                    .build();

            kinesisClient.createStream(streamReq);

        } catch (KinesisException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see CreateStream in *AWS SDK for Java 2.x API Reference*.

`DeleteStream`

The following code example shows how to use `DeleteStream`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DeleteStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataStream {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream (for example,
 StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        deleteStream(kinesisClient, streamName);
        kinesisClient.close();
        System.out.println("Done");
    }
```

```
    public static void deleteStream(KinesisClient kinesisClient, String streamName)
 {
        try {
            DeleteStreamRequest delStream = DeleteStreamRequest.builder()
                    .streamName(streamName)
                    .build();

            kinesisClient.deleteStream(delStream);

        } catch (KinesisException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteStream in *AWS SDK for Java 2.x API Reference*.

`GetRecords`

The following code example shows how to use `GetRecords`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.Shard;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorRequest;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorResponse;
import software.amazon.awssdk.services.kinesis.model.Record;
import software.amazon.awssdk.services.kinesis.model.GetRecordsRequest;
import software.amazon.awssdk.services.kinesis.model.GetRecordsResponse;
```

```
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetRecords {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream to read from (for
 example, StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        getStockTrades(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void getStockTrades(KinesisClient kinesisClient, String
 streamName) {
        String shardIterator;
        String lastShardId = null;
        DescribeStreamRequest describeStreamRequest =
 DescribeStreamRequest.builder()
```

```
                .streamName(streamName)
                .build();

        List<Shard> shards = new ArrayList<>();
        DescribeStreamResponse streamRes;
        do {
            streamRes = kinesisClient.describeStream(describeStreamRequest);
            shards.addAll(streamRes.streamDescription().shards());

            if (shards.size() > 0) {
                lastShardId = shards.get(shards.size() - 1).shardId();
            }
        } while (streamRes.streamDescription().hasMoreShards());

        GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
                .streamName(streamName)
                .shardIteratorType("TRIM_HORIZON")
                .shardId(lastShardId)
                .build();

        GetShardIteratorResponse shardIteratorResult =
 kinesisClient.getShardIterator(itReq);
        shardIterator = shardIteratorResult.shardIterator();

        // Continuously read data records from shard.
        List<Record> records;

        // Create new GetRecordsRequest with existing shardIterator.
        // Set maximum records to return to 1000.
        GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
                .shardIterator(shardIterator)
                .limit(1000)
                .build();

        GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

        // Put result into record list. Result may be empty.
        records = result.records();

        // Print records
        for (Record record : records) {
            SdkBytes byteBuffer = record.data();
            System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
 String(byteBuffer.asByteArray()));
```

```
        }
    }
}
```

- For API details, see GetRecords in *AWS SDK for Java 2.x API Reference*.

PutRecord

The following code example shows how to use `PutRecord`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>
```

```
                Where:
                    streamName - The Amazon Kinesis data stream to which records are
written (for example, StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        // Ensure that the Kinesis Stream is valid.
        validateStream(kinesisClient, streamName);
        setStockData(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void setStockData(KinesisClient kinesisClient, String streamName)
{
        try {
            // Repeatedly send stock trades with a 100 milliseconds wait in between.
            StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

            // Put in 50 Records for this example.
            int index = 50;
            for (int x = 0; x < index; x++) {
                StockTrade trade = stockTradeGenerator.getRandomTrade();
                sendStockTrade(trade, kinesisClient, streamName);
                Thread.sleep(100);
            }

        } catch (KinesisException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
```

```
    private static void sendStockTrade(StockTrade trade, KinesisClient
 kinesisClient,
            String streamName) {
        byte[] bytes = trade.toJsonAsBytes();

        // The bytes could be null if there is an issue with the JSON serialization
 by
        // the Jackson JSON library.
        if (bytes == null) {
            System.out.println("Could not get JSON bytes for stock trade");
            return;
        }

        System.out.println("Putting trade: " + trade);
        PutRecordRequest request = PutRecordRequest.builder()
                .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol
 as the partition key, explained in
                                                       // the Supplemental
 Information section below.
                .streamName(streamName)
                .data(SdkBytes.fromByteArray(bytes))
                .build();

        try {
            kinesisClient.putRecord(request);
        } catch (KinesisException e) {
            System.err.println(e.getMessage());
        }
    }

    private static void validateStream(KinesisClient kinesisClient, String
 streamName) {
        try {
            DescribeStreamRequest describeStreamRequest =
 DescribeStreamRequest.builder()
                    .streamName(streamName)
                    .build();

            DescribeStreamResponse describeStreamResponse =
 kinesisClient.describeStream(describeStreamRequest);

            if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
 {
```

```
                System.err.println("Stream " + streamName + " is not active. Please
 wait a few moments and try again.");
                System.exit(1);
            }

        } catch (KinesisException e) {
            System.err.println("Error found while describing the stream " +
 streamName);
            System.err.println(e);
            System.exit(1);
        }
    }
}
```

- For API details, see PutRecord in *AWS SDK for Java 2.x API Reference*.

Serverless examples

Invoke a Lambda function from a Kinesis trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving records from a Kinesis stream. The function retrieves the Kinesis payload, decodes from Base64, and logs the record contents.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming a Kinesis event with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }

}
```

Reporting batch item failures for Lambda functions with a Kinesis trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from a Kinesis stream. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Reporting Kinesis batch item failures with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
 input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse(batchItemFailures);
    }
}
```

AWS KMS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS KMS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello KMS key

The following code example shows how to get started using KMS key.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
                .region(region)
                .build();

        listAllKeys(kmsClient);
        kmsClient.close();
    }

    public static void listAllKeys(KmsClient kmsClient) {
        try {
            ListKeysRequest listKeysRequest = ListKeysRequest.builder()
                    .limit(15)
                    .build();

            ListKeysIterable keysResponse =
 kmsClient.listKeysPaginator(listKeysRequest);
            keysResponse.stream()
                .flatMap(r -> r.keys().stream())
                .forEach(key -> System.out
                    .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
 " + key.keyId()));

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see listKeysPaginator in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

CreateAlias

The following code example shows how to use `CreateAlias`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void createCustomAlias(KmsClient kmsClient, String targetKeyId, 
String aliasName) {
        try {
            CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
                .aliasName(aliasName)
                .targetKeyId(targetKeyId)
                .build();

            kmsClient.createAlias(aliasRequest);
            System.out.println(aliasName + " was successfully created.");

        } catch (ResourceExistsException e) {
            System.err.println("Alias already exists: " + e.getMessage());
            System.err.println("Moving on...");
        } catch (Exception e) {
            System.err.println("An unexpected error occurred: " + e.getMessage());
            System.err.println("Moving on...");
        }
    }
```

- For API details, see CreateAlias in *AWS SDK for Java 2.x API Reference*.

CreateGrant

The following code example shows how to use `CreateGrant`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/kms#code-examples).

```
    public static String grantKey(KmsClient kmsClient, String keyId, String
 granteePrincipal)  {
        try {
            // Add the desired KMS Grant permissions.
            List<GrantOperation> grantPermissions = new ArrayList<>();
            grantPermissions.add(GrantOperation.ENCRYPT);
            grantPermissions.add(GrantOperation.DECRYPT);
            grantPermissions.add(GrantOperation.DESCRIBE_KEY);

            CreateGrantRequest grantRequest = CreateGrantRequest.builder()
                .keyId(keyId)
                .name("grant1")
                .granteePrincipal(granteePrincipal)
                .operations(grantPermissions)
                .build();

            CreateGrantResponse response = kmsClient.createGrant(grantRequest);
            return response.grantId();

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see [CreateGrant](https://docs.aws.amazon.com/goto/SdkForJavaV2/kms-2014-11-01/CreateGrant) in *AWS SDK for Java 2.x API Reference*.

CreateKey

The following code example shows how to use `CreateKey`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createKey(KmsClient kmsClient, String keyDesc) {
        try {
            CreateKeyRequest keyRequest = CreateKeyRequest.builder()
                .description(keyDesc)
                .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
                .keyUsage("ENCRYPT_DECRYPT")
                .build();

            CreateKeyResponse result = kmsClient.createKey(keyRequest);
            System.out.println("Symmetric key with ARN [" +
 result.keyMetadata().arn() + "] has been created.");
            return result.keyMetadata().keyId();

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateKey in *AWS SDK for Java 2.x API Reference*.

`Decrypt`

The following code example shows how to use `Decrypt`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
 String keyId) {
        try {
            DecryptRequest decryptRequest = DecryptRequest.builder()
                .ciphertextBlob(encryptedData)
                .keyId(keyId)
                .build();

            DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
            return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see Decrypt in *AWS SDK for Java 2.x API Reference*.

`DeleteAlias`

The following code example shows how to use `DeleteAlias`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
        try {
            DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
                .aliasName(aliasName)
                .build();

            kmsClient.deleteAlias(deleteAliasRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteAlias in *AWS SDK for Java 2.x API Reference*.

DescribeKey

The following code example shows how to use `DescribeKey`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
        try {
            DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
                .keyId(keyId)
                .build();

            DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
            KeyState keyState = response.keyMetadata().keyState();
            if (keyState == KeyState.ENABLED) {
                System.out.println("The key is enabled.");
                return true;
            } else {
```

```
                System.out.println("The key is not enabled. Key state: " +
keyState);
            }

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return false;
    }
```

- For API details, see DescribeKey in *AWS SDK for Java 2.x API Reference*.

DisableKey

The following code example shows how to use `DisableKey`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void disableKey(KmsClient kmsClient, String keyId) {
        try {
            DisableKeyRequest keyRequest = DisableKeyRequest.builder()
                .keyId(keyId)
                .build();

            kmsClient.disableKey(keyRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DisableKey in *AWS SDK for Java 2.x API Reference*.

EnableKey

The following code example shows how to use `EnableKey`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Enable the KMS key.
    public static void enableKey(KmsClient kmsClient, String keyId) {
        try {
            EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
                .keyId(keyId)
                .build();

            kmsClient.enableKey(enableKeyRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see EnableKey in *AWS SDK for Java 2.x API Reference*.

Encrypt

The following code example shows how to use `Encrypt`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String 
 text) {
        try {
            SdkBytes myBytes = SdkBytes.fromUtf8String(text);
            EncryptRequest encryptRequest = EncryptRequest.builder()
                .keyId(keyId)
                .plaintext(myBytes)
                .build();

            EncryptResponse response = kmsClient.encrypt(encryptRequest);
            String algorithm = response.encryptionAlgorithm().toString();
            System.out.println("The string was encrypted with algorithm " + 
 algorithm + ".");

            // Get the encrypted data.
            SdkBytes encryptedData = response.ciphertextBlob();
            return encryptedData;

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return null;
    }
```

- For API details, see Encrypt in *AWS SDK for Java 2.x API Reference*.

ListAliases

The following code example shows how to use `ListAliases`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/kms#code-examples).

```
    public static void listAllAliases(KmsClient kmsClient) {
        try {
            ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
                .limit(15)
                .build();

            ListAliasesIterable aliasesResponse =
 kmsClient.listAliasesPaginator(aliasesRequest);
            aliasesResponse.stream()
                .flatMap(r -> r.aliases().stream())
                .forEach(alias -> System.out
                    .println("The alias name is: " + alias.aliasName()));

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListAliases in *AWS SDK for Java 2.x API Reference*.

`ListGrants`

The following code example shows how to use `ListGrants`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void displayGrantIds(KmsClient kmsClient, String keyId) {
        try {
            ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
                .keyId(keyId)
                .limit(15)
                .build();

            ListGrantsIterable response =
 kmsClient.listGrantsPaginator(grantsRequest);
            response.stream()
                .flatMap(r -> r.grants().stream())
                .forEach(grant -> {
                    System.out.println("The grant Id is : " + grant.grantId());
                    List<GrantOperation> ops = grant.operations();
                    for (GrantOperation op : ops) {
                        System.out.println(op.name());
                    }
                });

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListGrants in *AWS SDK for Java 2.x API Reference*.

`ListKeyPolicies`

The following code example shows how to use `ListKeyPolicies`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
 policyName) {
        try {
            GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
                .keyId(keyId)
                .policyName(policyName)
                .build();

            GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
            System.out.println("The response is "+response.policy());
        } catch (KmsException e) {
            if (e.getMessage().contains("No such policy exists")) {
                System.out.println("The policy cannot be found. Error message: " +
 e.getMessage());
            } else {
                throw e;
            }
        }
    }
```

- For API details, see [ListKeyPolicies](#) in *AWS SDK for Java 2.x API Reference*.

`ListKeys`

The following code example shows how to use `ListKeys`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
                .region(region)
                .build();

        listAllKeys(kmsClient);
        kmsClient.close();
    }

    public static void listAllKeys(KmsClient kmsClient) {
        try {
            ListKeysRequest listKeysRequest = ListKeysRequest.builder()
                    .limit(15)
                    .build();

            ListKeysIterable keysResponse =
 kmsClient.listKeysPaginator(listKeysRequest);
            keysResponse.stream()
                .flatMap(r -> r.keys().stream())
                .forEach(key -> System.out
                    .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
 " + key.keyId()));

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListKeys in *AWS SDK for Java 2.x API Reference*.

RevokeGrant

The following code example shows how to use RevokeGrant.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
 grantId) {
        try {
            RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
                .keyId(keyId)
                .grantId(grantId)
                .build();

            kmsClient.revokeGrant(grantRequest);
            System.out.println("Grant ID: [" + grantId +"] was successfully
 revoked!");

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see RevokeGrant in *AWS SDK for Java 2.x API Reference*.

ScheduleKeyDeletion

The following code example shows how to use ScheduleKeyDeletion.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteKey(KmsClient kmsClient, String keyId) {
        try {
            ScheduleKeyDeletionRequest deletionRequest =
 ScheduleKeyDeletionRequest.builder()
                .keyId(keyId)
                .pendingWindowInDays(7)
                .build();

            kmsClient.scheduleKeyDeletion(deletionRequest);
            System.out.println("The key will be deleted in 7 days.");

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ScheduleKeyDeletion in *AWS SDK for Java 2.x API Reference*.

Sign

The following code example shows how to use `Sign`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void signVerifyData(KmsClient kmsClient) {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest request = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048) // Specify key spec
            .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
            .origin(OriginType.AWS_KMS) // Specify key origin
            .build();

        CreateKeyResponse response = kmsClient.createKey(request);
        String keyId2 = response.keyMetadata().keyId();
        System.out.println("Created KMS key with ID: " + keyId2);

        SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
        SignRequest signRequest = SignRequest.builder()
            .keyId(keyId2)
            .message(bytes)
            .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
            .build();

        SignResponse signResponse = kmsClient.sign(signRequest);
        byte[] signedBytes = signResponse.signature().asByteArray();

        // Verify the digital signature.
        VerifyRequest verifyRequest = VerifyRequest.builder()
            .keyId(keyId2)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))
            .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
            .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
            .build();

        VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
        System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
    }
```

- For API details, see Sign in *AWS SDK for Java 2.x API Reference*.

TagResource

The following code example shows how to use `TagResource`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/kms#code-examples).

```
public static void tagKMSKey(KmsClient kmsClient, String keyId) {
    try {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        kmsClient.tagResource(tagResourceRequest);
        System.out.println("The key has been tagged.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [TagResource](https://docs.aws.amazon.com/goto/SdkForJavaV2/kms-2014-11-01/TagResource) in *AWS SDK for Java 2.x API Reference*.

Scenarios

Learn KMS key core operations

The following code example shows how to:

- Create a KMS key.
- List KMS keys for your account and get details about them.
- Enable and disable KMS keys.
- Generate a symmetric data key that can be used for client-side encryption.
- Generate an asymmetric key used to digitally sign data.
- Tag keys.
- Delete KMS keys.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.AliasListEntry;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.CreateAliasRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantResponse;
import software.amazon.awssdk.services.kms.model.CreateKeyRequest;
import software.amazon.awssdk.services.kms.model.CreateKeyResponse;
import software.amazon.awssdk.services.kms.model.CustomerMasterKeySpec;
import software.amazon.awssdk.services.kms.model.DecryptRequest;
import software.amazon.awssdk.services.kms.model.DecryptResponse;
import software.amazon.awssdk.services.kms.model.DeleteAliasRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyResponse;
import software.amazon.awssdk.services.kms.model.DisableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationRequest;
import software.amazon.awssdk.services.kms.model.EncryptRequest;
import software.amazon.awssdk.services.kms.model.EncryptResponse;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyResponse;
```

```
import software.amazon.awssdk.services.kms.model.GrantOperation;
import software.amazon.awssdk.services.kms.model.KeySpec;
import software.amazon.awssdk.services.kms.model.KeyState;
import software.amazon.awssdk.services.kms.model.KeyUsageType;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.LimitExceededException;
import software.amazon.awssdk.services.kms.model.ListAliasesRequest;
import software.amazon.awssdk.services.kms.model.ListGrantsRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesResponse;
import software.amazon.awssdk.services.kms.model.OriginType;
import software.amazon.awssdk.services.kms.model.PutKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.RevokeGrantRequest;
import software.amazon.awssdk.services.kms.model.ScheduleKeyDeletionRequest;
import software.amazon.awssdk.services.kms.model.SignRequest;
import software.amazon.awssdk.services.kms.model.SignResponse;
import software.amazon.awssdk.services.kms.model.SigningAlgorithmSpec;
import software.amazon.awssdk.services.kms.model.Tag;
import software.amazon.awssdk.services.kms.model.TagResourceRequest;
import software.amazon.awssdk.services.kms.model.VerifyRequest;
import software.amazon.awssdk.services.kms.model.VerifyResponse;
import software.amazon.awssdk.services.kms.paginators.ListAliasesIterable;
import software.amazon.awssdk.services.kms.paginators.ListGrantsIterable;
import software.amazon.awssdk.services.secretsmanager.model.ResourceExistsException;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;
import java.nio.ByteBuffer;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.ArrayList;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        final String usage = """
                Usage: <granteePrincipal>

                Where:
                   granteePrincipal - The principal (user, service account, or
 group) to whom the grant or permission is being given.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String granteePrincipal = args[0];
        String policyName = "default";

        Scanner scanner = new Scanner(System.in);
        String keyDesc = "Created by the AWS KMS API";

        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("""
            Welcome to the AWS Key Management SDK Getting Started scenario.

            This program demonstrates how to interact with AWS Key Management using
 the AWS SDK for Java (v2).
            The AWS Key Management Service (KMS) is a secure and highly available
 service that allows you to create
            and manage AWS KMS keys and control their use across a wide range of AWS
 services and applications.
            KMS provides a centralized and unified approach to managing encryption
 keys, making it easier to meet your
            data protection and regulatory compliance requirements.

            This Getting Started scenario creates two key types. A symmetric
 encryption key is used to encrypt and decrypt data,
            and an asymmetric key used to digitally sign data.
            Let's get started...
```

```
            """);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("1. Create a symmetric KMS key\n");
        System.out.println("First, the program will creates a symmetric KMS key that
 you can used to encrypt and decrypt data.");
        waitForInputToContinue(scanner);
        String targetKeyId = createKey(kmsClient, keyDesc);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            2. Enable a KMS key

            By default, when the SDK creates an AWS key it is enabled. The next bit
 of code checks to
            determine if the key is enabled. If it is not enabled, the code enables
 it.
             """);
        waitForInputToContinue(scanner);
        boolean isEnabled = isKeyEnabled(kmsClient, targetKeyId);
        if (!isEnabled)
            enableKey(kmsClient, targetKeyId);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        System.out.printf("""

                One of the main uses of symmetric keys is to encrypt and decrypt
 data.
                Next, the code encrypts the string '%s' with the SYMMETRIC_DEFAULT
 encryption algorithm.
                %n""", plaintext);
        waitForInputToContinue(scanner);
        SdkBytes ciphertext = encryptData(kmsClient, targetKeyId, plaintext);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("4. Create an alias");
        System.out.println("""
```

```
            Enter an alias name for the key. The name should be prefixed with
'alias/'.
            For example, 'alias/myFirstKey'.
            """);

        String aliasName = scanner.nextLine();
        String fullAliasName = aliasName.isEmpty() ? "alias/dev-encryption-key" :
aliasName;
        createCustomAlias(kmsClient, targetKeyId, fullAliasName);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("5. List all of your aliases");
        waitForInputToContinue(scanner);
        listAllAliases(kmsClient);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("6. Enable automatic rotation of the KMS key");
        System.out.println("""

            By default, when the SDK enables automatic rotation of a KMS key,
            KMS rotates the key material of the KMS key one year (approximately 365
days) from the enable date and every year
            thereafter.
            """);
        waitForInputToContinue(scanner);
        enableKeyRotation(kmsClient, targetKeyId);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            7. Create a grant

            A grant is a policy instrument that allows Amazon Web Services
principals to use KMS keys.
            It also can allow them to view a KMS key (DescribeKey) and create and
manage grants.
            When authorizing access to a KMS key, grants are considered along with
key policies and IAM policies.
            """);

        waitForInputToContinue(scanner);
        String grantId = grantKey(kmsClient, targetKeyId, granteePrincipal);
```

```
        System.out.println("The code granted principal with ARN [" +
granteePrincipal + "] ");
        System.out.println("use of the symmetric key. The grant ID is [" + grantId +
"]");
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("8. List grants for the KMS key");
        waitForInputToContinue(scanner);
        displayGrantIds(kmsClient, targetKeyId);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("9. Revoke the grant");
        waitForInputToContinue(scanner);
        revokeKeyGrant(kmsClient, targetKeyId, grantId);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("10. Decrypt the data\n");
        System.out.println("""
            Lets decrypt the data that was encrypted in an early step.
            The code uses the same key to decrypt the string that we encrypted
earlier in the program.
            """);
        waitForInputToContinue(scanner);
        String decryptText = decryptData(kmsClient, ciphertext, targetKeyId);
        System.out.println("Decrypted text is: " + decryptText);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("11. Replace a key policy\n");
        System.out.println("""
            A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
            access to KMS keys. Every KMS key must have exactly one key policy. The
statements in the key policy
            determine who has permission to use the KMS key and how they can use
it.
            You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
            must have a key policy.
```

```
            By default, when you create a key by using the SDK, a policy is created
that
            gives the AWS account that owns the KMS key full access to the KMS key.

            Let's try to replace the automatically created policy with the following
policy.

                "Version": "2012-10-17",
                "Statement": [{
                "Effect": "Allow",
                "Principal": {"AWS": "arn:aws:iam::0000000000:root"},
                "Action": "kms:*",
                "Resource": "*"
                }]
            """);

        waitForInputToContinue(scanner);
        boolean polAdded = replacePolicy(kmsClient, targetKeyId, policyName);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("12. Get the key policy\n");
        System.out.println("The next bit of code that runs gets the key policy to
make sure it exists.");
        waitForInputToContinue(scanner);
        getKeyPolicy(kmsClient, targetKeyId, policyName);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("13. Create an asymmetric KMS key and sign your data\n");
        System.out.println("""
            Signing your data with an AWS key can provide several benefits that make
it an attractive option
            for your data signing needs. By using an AWS KMS key, you can leverage
the
            security controls and compliance features provided by AWS,
            which can help you meet various regulatory requirements and enhance the
overall security posture
            of your organization.
           """);
        waitForInputToContinue(scanner);
        signVerifyData(kmsClient);
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);
        System.out.println("14. Tag your symmetric KMS Key\n");
        System.out.println("""
            By using tags, you can improve the overall management, security, and
governance of your
            KMS keys, making it easier to organize, track, and control access to
your encrypted data within
            your AWS environment
            """);
        waitForInputToContinue(scanner);
        tagKMSKey(kmsClient, targetKeyId);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("15. Schedule the deletion of the KMS key\n");
        System.out.println("""
            By default, KMS applies a waiting period of 30 days,
            but you can specify a waiting period of 7-30 days. When this operation
is successful,
            the key state of the KMS key changes to PendingDeletion and the key
can't be used in any
            cryptographic operations. It remains in this state for the duration of
the waiting period.

            Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
            all data that was encrypted under the KMS key is unrecoverable.\s
            """);
        System.out.println("Would you like to delete the Key Management resources?
(y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the AWS KMS resources.");
            waitForInputToContinue(scanner);
            deleteSpecificAlias(kmsClient, fullAliasName);
            disableKey(kmsClient, targetKeyId);
            deleteKey(kmsClient, targetKeyId);

        } else {
            System.out.println("The Key Management resources will not be deleted");
        }

        System.out.println(DASHES);
```

```
        System.out.println("This concludes the AWS Key Management SDK Getting
 Started scenario");
        System.out.println(DASHES);
    }
    public static void listAllAliases(KmsClient kmsClient) {
        try {
            ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
                .limit(15)
                .build();

            ListAliasesIterable aliasesResponse =
 kmsClient.listAliasesPaginator(aliasesRequest);
            aliasesResponse.stream()
                .flatMap(r -> r.aliases().stream())
                .forEach(alias -> System.out
                    .println("The alias name is: " + alias.aliasName()));

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void disableKey(KmsClient kmsClient, String keyId) {
        try {
            DisableKeyRequest keyRequest = DisableKeyRequest.builder()
                .keyId(keyId)
                .build();

            kmsClient.disableKey(keyRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void signVerifyData(KmsClient kmsClient) {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest request = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048) // Specify key spec
            .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
```

```
            .origin(OriginType.AWS_KMS) // Specify key origin
            .build();

        CreateKeyResponse response = kmsClient.createKey(request);
        String keyId2 = response.keyMetadata().keyId();
        System.out.println("Created KMS key with ID: " + keyId2);

        SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
        SignRequest signRequest = SignRequest.builder()
            .keyId(keyId2)
            .message(bytes)
            .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
            .build();

        SignResponse signResponse = kmsClient.sign(signRequest);
        byte[] signedBytes = signResponse.signature().asByteArray();

        // Verify the digital signature.
        VerifyRequest verifyRequest = VerifyRequest.builder()
            .keyId(keyId2)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))
            .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
            .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
            .build();

        VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
        System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
    }

    public static void tagKMSKey(KmsClient kmsClient, String keyId) {
        try {
            Tag tag = Tag.builder()
                .tagKey("Environment")
                .tagValue("Production")
                .build();

            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                .keyId(keyId)
                .tags(tag)
                .build();

            kmsClient.tagResource(tagResourceRequest);
```

```
            System.out.println("The key has been tagged.");

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
 policyName) {
        try {
            GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
                .keyId(keyId)
                .policyName(policyName)
                .build();

            GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
            System.out.println("The response is "+response.policy());
        } catch (KmsException e) {
            if (e.getMessage().contains("No such policy exists")) {
                System.out.println("The policy cannot be found. Error message: " +
 e.getMessage());
            } else {
                throw e;
            }
        }
    }

    public static boolean replacePolicy(KmsClient kmsClient, String keyId, String
 policyName) {
        // Change the principle in the below JSON.
        String policy = """
            {
              "Version": "2012-10-17",
              "Statement": [{
                "Effect": "Allow",
                "Principal": {"AWS": "arn:aws:iam::%s:root"},
                "Action": "kms:*",
                "Resource": "*"
              }]
            }
            """.formatted(accountId);

        try {
```

```
            PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
                .keyId(keyId)
                .policyName(policyName)
                .policy(policy)
                .build();
            kmsClient.putKeyPolicy(keyPolicyRequest);
            System.out.println("The key policy has been replaced.");
        } catch (LimitExceededException e) {
            System.out.println("Policy limit reached. Unable to create the
 policy.");
            return false;
        } catch (AlreadyExistsException e) {
            System.out.println("Only one policy per key is supported. Unable to
 create the policy.");
            return false;
        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        return true;
    }

    public static boolean doesKeyHavePolicy(KmsClient kmsClient, String keyId,
 String policyName){
        ListKeyPoliciesRequest policiesRequest = ListKeyPoliciesRequest.builder()
            .keyId(keyId)
            .build();

        boolean hasPolicy = false;
        ListKeyPoliciesResponse response =
 kmsClient.listKeyPolicies(policiesRequest);
        List<String>policyNames = response.policyNames();
        for (String pol : policyNames) {
            hasPolicy = true;
        }
        return hasPolicy;
    }

    public static void deleteKey(KmsClient kmsClient, String keyId) {
        try {
            ScheduleKeyDeletionRequest deletionRequest =
 ScheduleKeyDeletionRequest.builder()
                .keyId(keyId)
```

```
                .pendingWindowInDays(7)
                .build();

            kmsClient.scheduleKeyDeletion(deletionRequest);
            System.out.println("The key will be deleted in 7 days.");

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
        try {
            DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
                .aliasName(aliasName)
                .build();

            kmsClient.deleteAlias(deleteAliasRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
        try {
            DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
                .keyId(keyId)
                .build();

            DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
            KeyState keyState = response.keyMetadata().keyState();
            if (keyState == KeyState.ENABLED) {
                System.out.println("The key is enabled.");
                return true;
            } else {
                System.out.println("The key is not enabled. Key state: " +
 keyState);
            }

        } catch (KmsException e) {
            System.err.println(e.getMessage());
```

```
            System.exit(1);
        }
        return false;
    }

    public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
 String keyId) {
        try {
            DecryptRequest decryptRequest = DecryptRequest.builder()
                .ciphertextBlob(encryptedData)
                .keyId(keyId)
                .build();

            DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
            return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
 grantId) {
        try {
            RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
                .keyId(keyId)
                .grantId(grantId)
                .build();

            kmsClient.revokeGrant(grantRequest);
            System.out.println("Grant ID: [" + grantId +"] was successfully
 revoked!");

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void displayGrantIds(KmsClient kmsClient, String keyId) {
        try {
            ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
```

```
                .keyId(keyId)
                .limit(15)
                .build();

            ListGrantsIterable response =
 kmsClient.listGrantsPaginator(grantsRequest);
            response.stream()
                .flatMap(r -> r.grants().stream())
                .forEach(grant -> {
                    System.out.println("The grant Id is : " + grant.grantId());
                    List<GrantOperation> ops = grant.operations();
                    for (GrantOperation op : ops) {
                        System.out.println(op.name());
                    }
                });

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String grantKey(KmsClient kmsClient, String keyId, String
 granteePrincipal)  {
        try {
            // Add the desired KMS Grant permissions.
            List<GrantOperation> grantPermissions = new ArrayList<>();
            grantPermissions.add(GrantOperation.ENCRYPT);
            grantPermissions.add(GrantOperation.DECRYPT);
            grantPermissions.add(GrantOperation.DESCRIBE_KEY);

            CreateGrantRequest grantRequest = CreateGrantRequest.builder()
                .keyId(keyId)
                .name("grant1")
                .granteePrincipal(granteePrincipal)
                .operations(grantPermissions)
                .build();

            CreateGrantResponse response = kmsClient.createGrant(grantRequest);
            return response.grantId();

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
        return "";
    }

    public static void enableKeyRotation(KmsClient kmsClient, String keyId) {
        try {
            EnableKeyRotationRequest enableKeyRotationRequest =
 EnableKeyRotationRequest.builder()
                .keyId(keyId)
                .build();

            kmsClient.enableKeyRotation(enableKeyRotationRequest);
            System.out.println("Key rotation has been enabled for key with id [" +
 keyId + "]");

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
 String aliasName) {
        try {
            CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
                .aliasName(aliasName)
                .targetKeyId(targetKeyId)
                .build();

            kmsClient.createAlias(aliasRequest);
            System.out.println(aliasName + " was successfully created.");

        } catch (ResourceExistsException e) {
            System.err.println("Alias already exists: " + e.getMessage());
            System.err.println("Moving on...");
        } catch (Exception e) {
            System.err.println("An unexpected error occurred: " + e.getMessage());
            System.err.println("Moving on...");
        }
    }

    public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String
 text) {
        try {
```

```
            SdkBytes myBytes = SdkBytes.fromUtf8String(text);
            EncryptRequest encryptRequest = EncryptRequest.builder()
                .keyId(keyId)
                .plaintext(myBytes)
                .build();

            EncryptResponse response = kmsClient.encrypt(encryptRequest);
            String algorithm = response.encryptionAlgorithm().toString();
            System.out.println("The string was encrypted with algorithm " +
 algorithm + ".");

            // Get the encrypted data.
            SdkBytes encryptedData = response.ciphertextBlob();
            return encryptedData;

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return null;
    }

    public static String createKey(KmsClient kmsClient, String keyDesc) {
        try {
            CreateKeyRequest keyRequest = CreateKeyRequest.builder()
                .description(keyDesc)
                .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
                .keyUsage("ENCRYPT_DECRYPT")
                .build();

            CreateKeyResponse result = kmsClient.createKey(keyRequest);
            System.out.println("Symmetric key with ARN [" +
 result.keyMetadata().arn() + "] has been created.");
            return result.keyMetadata().keyId();

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Enable the KMS key.
    public static void enableKey(KmsClient kmsClient, String keyId) {
```

```
        try {
            EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
                .keyId(keyId)
                .build();

            kmsClient.enableKey(enableKeyRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
    private static String getAccountId(){
        try (StsClient stsClient = StsClient.create()){
            GetCallerIdentityResponse callerIdentity =
 stsClient.getCallerIdentity();
            return callerIdentity.account();
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateKey
  - DescribeKey
  - DisableKey

- EnableKey
- GenerateDataKey
- ListKeys
- ScheduleKeyDeletion
- Sign

Lambda examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Lambda.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Lambda

The following code examples show how to get started using Lambda.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package com.example.lambda;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
```

```
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLambdaFunctions {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
                .region(region)
                .build();

        listFunctions(awsLambda);
        awsLambda.close();
    }

    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListFunctions in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios
- Serverless examples

Actions

CreateFunction

The following code example shows how to use `CreateFunction`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.CreateFunctionRequest;
import software.amazon.awssdk.services.lambda.model.FunctionCode;
import software.amazon.awssdk.services.lambda.model.CreateFunctionResponse;
import software.amazon.awssdk.services.lambda.model.GetFunctionRequest;
import software.amazon.awssdk.services.lambda.model.GetFunctionResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.Runtime;
import software.amazon.awssdk.services.lambda.waiters.LambdaWaiter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

/**
 * This code example requires a ZIP or JAR that represents the code of the
 * Lambda function.
 * If you do not have a ZIP or JAR, please refer to the following document:
 *
 * https://github.com/aws-doc-sdk-examples/tree/master/javav2/usecases/
creating_workflows_stepfunctions
```

```
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateFunction {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <functionName> <filePath> <role> <handler>\s

                Where:
                    functionName - The name of the Lambda function.\s
                    filePath - The path to the ZIP or JAR where the code is located.
\s
                    role - The role ARN that has Lambda permissions.\s
                    handler - The fully qualified method name (for example,
 example.Handler::handleRequest). \s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        String role = args[2];
        String handler = args[3];
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
                .region(region)
                .build();

        createLambdaFunction(awsLambda, functionName, filePath, role, handler);
        awsLambda.close();
    }

    public static void createLambdaFunction(LambdaClient awsLambda,
```

```
            String functionName,
            String filePath,
            String role,
            String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

            FunctionCode code = FunctionCode.builder()
                    .zipFile(fileToUpload)
                    .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                    .functionName(functionName)
                    .description("Created by the Lambda Java API")
                    .code(code)
                    .handler(handler)
                    .runtime(Runtime.JAVA8)
                    .role(role)
                    .build();

            // Create a Lambda function using a waiter.
            CreateFunctionResponse functionResponse =
 awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
 waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The function ARN is " +
 functionResponse.functionArn());

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see CreateFunction in *AWS SDK for Java 2.x API Reference*.

DeleteFunction

The following code example shows how to use `DeleteFunction`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/lambda#code-examples).

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFunction {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <functionName>\s

                Where:
                    functionName - The name of the Lambda function.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        Region region = Region.US_EAST_1;
```

```
        LambdaClient awsLambda = LambdaClient.builder()
                .region(region)
                .build();

        deleteLambdaFunction(awsLambda, functionName);
        awsLambda.close();
    }

    public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteFunction in *AWS SDK for Java 2.x API Reference*.

Invoke

The following code example shows how to use `Invoke`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import org.json.JSONObject;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;

public class LambdaInvoke {

    /*
     * Function names appear as
     * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
     * you can retrieve the value by looking at the function in the AWS Console
     *
     * Also, set up your development environment, including your credentials.
     *
     * For information, see this documentation topic:
     *
     * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.
     * html
     */

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <functionName>\s

                Where:
                    functionName - The name of the Lambda function\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
                .region(region)
                .build();

        invokeFunction(awsLambda, functionName);
```

```
        awsLambda.close();
    }

    public static void invokeFunction(LambdaClient awsLambda, String functionName) {

        InvokeResponse res = null;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            // Setup an InvokeRequest.
            InvokeRequest request = InvokeRequest.builder()
                    .functionName(functionName)
                    .payload(payload)
                    .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see Invoke in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with functions

The following code example shows how to:

- Create an IAM role and Lambda function, then upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure with an environment variable.

- Invoke the function with new parameters and get results. Display the returned execution log.
- List the functions for your account, then clean up resources.

For more information, see Create a Lambda function with the console.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/*
 *  Lambda function names appear as:
 *
 *  arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 *  To find this value, look at the function in the AWS Management Console.
 *
 *  Before running this Java code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see this documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *  This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <functionName> <filePath> <role> <handler> <bucketName> <key>\s

                Where:
                    functionName - The name of the Lambda function.\s
                    filePath - The path to the .zip or .jar where the code is
 located.\s
                    role - The AWS Identity and Access Management (IAM) service role
 that has Lambda permissions.\s
                    handler - The fully qualified method name (for example,
 example.Handler::handleRequest).\s
                    bucketName - The Amazon Simple Storage Service (Amazon S3)
 bucket name that contains the .zip or .jar used to update the Lambda function's
 code.\s
                    key - The Amazon S3 key name that represents the .zip or .jar
 (for example, LambdaHello-1.0-SNAPSHOT.jar).
                    """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        String role = args[2];
        String handler = args[3];
        String bucketName = args[4];
        String key = args[5];

        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Lambda example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an AWS Lambda function.");
```

```
        String funArn = createLambdaFunction(awsLambda, functionName, filePath,
 role, handler);
        System.out.println("The AWS Lambda ARN is " + funArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get the " + functionName + " AWS Lambda function.");
        getFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List all AWS Lambda functions.");
        listFunctions(awsLambda);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Invoke the Lambda function.");
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update the Lambda function code and invoke it
 again.");
        updateFunctionCode(awsLambda, functionName, bucketName, key);
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update a Lambda function's configuration value.");
        updateFunctionConfiguration(awsLambda, functionName, handler);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the AWS Lambda function.");
        LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Lambda scenario completed successfully");
        System.out.println(DASHES);
```

```
        awsLambda.close();
    }

    public static String createLambdaFunction(LambdaClient awsLambda,
            String functionName,
            String filePath,
            String role,
            String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

            FunctionCode code = FunctionCode.builder()
                    .zipFile(fileToUpload)
                    .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                    .functionName(functionName)
                    .description("Created by the Lambda Java API")
                    .code(code)
                    .handler(handler)
                    .runtime(Runtime.JAVA8)
                    .role(role)
                    .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
 awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
 waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

```
    public static void getFunction(LambdaClient awsLambda, String functionName) {
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            GetFunctionResponse response = awsLambda.getFunction(functionRequest);
            System.out.println("The runtime of this Lambda function is " +
 response.configuration().runtime());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void invokeFunction(LambdaClient awsLambda, String functionName) {

        InvokeResponse res;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            InvokeRequest request = InvokeRequest.builder()
                    .functionName(functionName)
                    .payload(payload)
```

```
                    .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateFunctionCode(LambdaClient awsLambda, String
 functionName, String bucketName, String key) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            UpdateFunctionCodeRequest functionCodeRequest =
 UpdateFunctionCodeRequest.builder()
                    .functionName(functionName)
                    .publish(true)
                    .s3Bucket(bucketName)
                    .s3Key(key)
                    .build();

            UpdateFunctionCodeResponse response =
 awsLambda.updateFunctionCode(functionCodeRequest);
            GetFunctionConfigurationRequest getFunctionConfigRequest =
 GetFunctionConfigurationRequest.builder()
                    .functionName(functionName)
                    .build();

            WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
                    .waitUntilFunctionUpdated(getFunctionConfigRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The last modified value is " +
 response.lastModified());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
    public static void updateFunctionConfiguration(LambdaClient awsLambda, String
 functionName, String handler) {
        try {
            UpdateFunctionConfigurationRequest configurationRequest =
 UpdateFunctionConfigurationRequest.builder()
                    .functionName(functionName)
                    .handler(handler)
                    .runtime(Runtime.JAVA11)
                    .build();

            awsLambda.updateFunctionConfiguration(configurationRequest);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke

- ListFunctions
- UpdateFunctionCode
- UpdateFunctionConfiguration

Serverless examples

Connecting to an Amazon RDS database in a Lambda function

The following code example shows how to implement a Lambda function that connects to an RDS database. The function makes a simple database request and returns the result.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Connecting to an Amazon RDS database in a Lambda function using Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements RequestHandler<APIGatewayProxyRequestEvent,
 APIGatewayProxyResponseEvent> {

    @Override
```

```
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new APIGatewayProxyResponseEvent();

        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"), token);
                 PreparedStatement statement = connection.prepareStatement("SELECT ?
 + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }

    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
```

```
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();

        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response = rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

Invoke a Lambda function from a Kinesis trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving records from a Kinesis stream. The function retrieves the Kinesis payload, decodes from Base64, and logs the record contents.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming a Kinesis event with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;
```

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }

}
```

Invoke a Lambda function from a DynamoDB trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving records from a DynamoDB stream. The function retrieves the DynamoDB payload and logs the record contents.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming a DynamoDB event with Lambda using Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import
 com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " + GSON.toJson(record.getDynamodb()));
    }
}
```

Invoke a Lambda function from an Amazon S3 trigger

The following code example shows how to implement a Lambda function that receives an event triggered by uploading an object to an S3 bucket. The function retrieves the S3 bucket name and object key from the event parameter and calls the Amazon S3 API to retrieve and log the content type of the object.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an S3 event with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);
```

```
            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

Invoke a Lambda function from an Amazon SNS trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving messages from an SNS topic. The function retrieves the messages from the event parameter and logs the content of each message.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an SNS event with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
```

```
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

Invoke a Lambda function from an Amazon SQS trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving messages from an SQS queue. The function retrieves the messages from the event parameter and logs the content of each message.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](https://github.com/aws-samples/serverless-snippets) repository.

Consuming an SQS event with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
```

```
        }

    }
}
```

Reporting batch item failures for Lambda functions with a Kinesis trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from a Kinesis stream. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Reporting Kinesis batch item failures with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";
```

```
        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
 input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse(batchItemFailures);
    }
}
```

Reporting batch item failures for Lambda functions with a DynamoDB trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from a DynamoDB stream. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Reporting DynamoDB batch item failures with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 Serializable> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
 {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
}
```

Reporting batch item failures for Lambda functions with an Amazon SQS trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from an SQS queue. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Reporting SQS batch item failures with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
 SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

         List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<SQSBatchResponse.BatchItemFailure>();
         String messageId = "";
         for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
             try {
                 //process your message
                 messageId = message.getMessageId();
             } catch (Exception e) {
```

```
                    //Add failed message identifier to the batchItemFailures list
                    batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(messageId));
                }
            }
            return new SQSBatchResponse(batchItemFailures);
        }
}
```

MediaConvert examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with MediaConvert.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

Actions

`CreateJob`

The following code example shows how to use `CreateJob`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
```

```
import
 software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
 software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
 software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import
 software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
```

```
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class CreateJob {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                    <mcRoleARN> <fileInput>\s

                                Where:
                                    mcRoleARN - The MediaConvert Role ARN.\s
                                    fileInput -  The URL of an Amazon S3 bucket
 where the input file is located.\s
                                """;
```

```
        if (args.length != 2) {
                System.out.println(usage);
                System.exit(1);
        }

        String mcRoleARN = args[0];
        String fileInput = args[1];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                        .region(region)
                        .build();

        String id = createMediaJob(mc, mcRoleARN, fileInput);
        System.out.println("MediaConvert job created. Job Id = " + id);
        mc.close();
    }

    public static String createMediaJob(MediaConvertClient mc, String mcRoleARN,
 String fileInput) {

        String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') +
 1) + "javasdk/out/";
        String fileOutput = s3path + "index";
        String thumbsOutput = s3path + "thumbs/";
        String mp4Output = s3path + "mp4/";

        try {
                DescribeEndpointsResponse res = mc

 .describeEndpoints(DescribeEndpointsRequest.builder().maxResults(20).build());

                if (res.endpoints().size() <= 0) {
                        System.out.println("Cannot find MediaConvert service
 endpoint URL!");
                        System.exit(1);
                }
                String endpointURL = res.endpoints().get(0).url();
                System.out.println("MediaConvert service URL: " +
 endpointURL);
                System.out.println("MediaConvert role arn: " + mcRoleARN);
                System.out.println("MediaConvert input file: " + fileInput);
                System.out.println("MediaConvert output path: " + s3path);

                MediaConvertClient emc = MediaConvertClient.builder()
```

```
                                         .region(Region.US_WEST_2)
                                         .endpointOverride(URI.create(endpointURL))
                                         .build();

                         // output group Preset HLS low profile
                         Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
 750000, 7, 1920, 1080, 640);
                         // output group Preset HLS media profile
                         Output hlsMedium = createOutput("hls_medium", "_medium", "_
$dt$", 1200000, 7, 1920, 1080, 1280);
                         // output group Preset HLS high profole
                         Output hlsHigh = createOutput("hls_high", "_high", "_$dt$",
 3500000, 8, 1920, 1080, 1920);

                         OutputGroup appleHLS = OutputGroup.builder().name("Apple
 HLS").customName("Example")

 .outputGroupSettings(OutputGroupSettings.builder()

 .type(OutputGroupType.HLS_GROUP_SETTINGS)

 .hlsGroupSettings(HlsGroupSettings.builder()

 .directoryStructure(

    HlsDirectoryStructure.SINGLE_DIRECTORY)

 .manifestDurationFormat(

    HlsManifestDurationFormat.INTEGER)

 .streamInfResolution(

    HlsStreamInfResolution.INCLUDE)

 .clientCache(HlsClientCache.ENABLED)

 .captionLanguageSetting(

    HlsCaptionLanguageSetting.OMIT)

 .manifestCompression(

    HlsManifestCompression.NONE)
```

```
.codecSpecification(

   HlsCodecSpecification.RFC_4281)

.outputSelection(

   HlsOutputSelection.MANIFESTS_AND_SEGMENTS)

.programDateTime(HlsProgramDateTime.EXCLUDE)

.programDateTimePeriod(600)

.timedMetadataId3Frame(

   HlsTimedMetadataId3Frame.PRIV)

.timedMetadataId3Period(10)

.destination(fileOutput)

.segmentControl(HlsSegmentControl.SEGMENTED_FILES)

.minFinalSegmentLength((double) 0)

.segmentLength(4).minSegmentLength(0).build())
                                                        .build())
                                        .outputs(hlsLow, hlsMedium,
hlsHigh).build();

                       OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(mp4Output).build())
                                                        .build())
                                        .outputs(Output.builder().extension("mp4")

.containerSettings(ContainerSettings.builder()
```

```
.container(ContainerType.MP4).build())

.videoDescription(VideoDescription.builder().width(1280)
                                                                                    .height(720)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

   VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

   .codec(VideoCodec.H_264)

   .h264Settings(H264Settings

                   .builder()

                   .rateControlMode(

                                  H264RateControlMode.QVBR)

                   .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)

                   .qualityTuningLevel(

                                  H264QualityTuningLevel.SINGLE_PASS)

                   .qvbrSettings(

                                  H264QvbrSettings.builder()
```

```
                                                .qvbrQualityLevel(
                                                                8)
                                                .build())
                        .codecLevel(H264CodecLevel.AUTO)
                        .codecProfile(H264CodecProfile.MAIN)
                        .maxBitrate(2400000)
                        .framerateControl(
                                        H264FramerateControl.INITIALIZE_FROM_SOURCE)
                        .gopSize(2.0)
                        .gopSizeUnits(H264GopSizeUnits.SECONDS)
                        .numberBFramesBetweenReferenceFrames(
                                        2)
                        .gopClosedCadence(
                                        1)
                        .gopBReference(H264GopBReference.DISABLED)
                        .slowPal(H264SlowPal.DISABLED)
                        .syntax(H264Syntax.DEFAULT)
                        .numberReferenceFrames(
                                        3)
                        .dynamicSubGop(H264DynamicSubGop.STATIC)
                        .fieldEncoding(H264FieldEncoding.PAFF)
                        .sceneChangeDetect(
```

```
                                H264SceneChangeDetect.ENABLED)
                .minIInterval(0)
                .telecine(H264Telecine.NONE)
                .framerateConversionAlgorithm(
                                H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                .entropyEncoding(
                                H264EntropyEncoding.CABAC)
                .slices(1)
                .unregisteredSeiTimecode(
                                H264UnregisteredSeiTimecode.DISABLED)
                .repeatPps(H264RepeatPps.DISABLED)
                .adaptiveQuantization(
                                H264AdaptiveQuantization.HIGH)
                .spatialAdaptiveQuantization(
                                H264SpatialAdaptiveQuantization.ENABLED)
                .temporalAdaptiveQuantization(
                                H264TemporalAdaptiveQuantization.ENABLED)
                .flickerAdaptiveQuantization(
                                H264FlickerAdaptiveQuantization.DISABLED)
                .softness(0)
                .interlaceMode(H264InterlaceMode.PROGRESSIVE)
                .build())
```

```
   .build())
                                                                        .build())

.audioDescriptions(AudioDescription.builder()

.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

.languageCodeControl(

   AudioLanguageCodeControl.FOLLOW_INPUT)

.codecSettings(AudioCodecSettings.builder()

   .codec(AudioCodec.AAC)

   .aacSettings(AacSettings

                   .builder()

                   .codecProfile(AacCodecProfile.LC)

                   .rateControlMode(

                                   AacRateControlMode.CBR)

                   .codingMode(AacCodingMode.CODING_MODE_2_0)

                   .sampleRate(44100)

                   .bitrate(160000)

                   .rawFormat(AacRawFormat.NONE)

                   .specification(AacSpecification.MPEG4)

                   .audioDescriptionBroadcasterMix(

                                   AacAudioDescriptionBroadcasterMix.NORMAL)

                   .build())

   .build())
                                                                        .build())
```

```
                                                                .build())
                                                .build();
                        OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(thumbsOutput).build())
                                                                .build())
                                                .outputs(Output.builder().extension("jpg")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.RAW).build())

.videoDescription(VideoDescription.builder()

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

   VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

   .codec(VideoCodec.FRAME_CAPTURE)

   .frameCaptureSettings(

                   FrameCaptureSettings

                                   .builder()

                                   .framerateNumerator(
```

```
                                                        1)

                                    .framerateDenominator(

                                                        1)

                                    .maxCaptures(10000000)

                                    .quality(80)

                                    .build())

    .build())
                                                                    .build())
                                                    .build())
                                    .build();

                    Map<String, AudioSelector> audioSelectors = new HashMap<>();
                    audioSelectors.put("Audio Selector 1",

AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
                                                    .offset(0).build());

                    JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
                                    .audioSelectors(audioSelectors)
                                    .videoSelector(

VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)

.rotate(InputRotate.DEGREE_0).build())

.filterEnable(InputFilterEnable.AUTO).filterStrength(0)
                                    .deblockFilter(InputDeblockFilter.DISABLED)

.denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)

.timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
                                    .outputGroups(appleHLS, thumbs,
fileMp4).build();

                    CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
```

```
                                        .settings(jobSettings)
                                        .build();

                        CreateJobResponse createJobResponse =
emc.createJob(createJobRequest);
                        return createJobResponse.job().id();

                } catch (MediaConvertException e) {
                        System.out.println(e.toString());
                        System.exit(0);
                }
                return "";
        }

        private final static Output createOutput(String customName,
                        String nameModifier,
                        String segmentModifier,
                        int qvbrMaxBitrate,
                        int qvbrQualityLevel,
                        int originWidth,
                        int originHeight,
                        int targetWidth) {

                int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
                                - (Math.round(originHeight * targetWidth /
originWidth) % 4);
                Output output = null;
                try {
                        output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
                                        .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)
```

```
.pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,

   489, 490, 491, 492)
                                                                                 .build())
                                                                 .build())
                                                 .videoDescription(

VideoDescription.builder().width(targetWidth)

.height(targetHeight)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

   VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

   .codec(VideoCodec.H_264)
```

```
    .h264Settings(H264Settings
                    .builder()
                    .rateControlMode(
                                    H264RateControlMode.QVBR)
                    .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                    .qualityTuningLevel(
                                    H264QualityTuningLevel.SINGLE_PASS)
                    .qvbrSettings(H264QvbrSettings
                                    .builder()
                                    .qvbrQualityLevel(
                                                    qvbrQualityLevel)
                                    .build())
                    .codecLevel(H264CodecLevel.AUTO)
                    .codecProfile((targetHeight > 720
                                    && targetWidth > 1280)
                                                    ? H264CodecProfile.HIGH
                                                    : H264CodecProfile.MAIN)
                    .maxBitrate(qvbrMaxBitrate)
                    .framerateControl(
                                    H264FramerateControl.INITIALIZE_FROM_SOURCE)
                    .gopSize(2.0)
                    .gopSizeUnits(H264GopSizeUnits.SECONDS)
```

```
                                .numberBFramesBetweenReferenceFrames(
                                                2)
                                .gopClosedCadence(
                                                1)
                                .gopBReference(H264GopBReference.DISABLED)
                                .slowPal(H264SlowPal.DISABLED)
                                .syntax(H264Syntax.DEFAULT)
                                .numberReferenceFrames(
                                                3)
                                .dynamicSubGop(H264DynamicSubGop.STATIC)
                                .fieldEncoding(H264FieldEncoding.PAFF)
                                .sceneChangeDetect(
                                                H264SceneChangeDetect.ENABLED)
                                .minIInterval(0)
                                .telecine(H264Telecine.NONE)
                                .framerateConversionAlgorithm(
                                                H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                                .entropyEncoding(
                                                H264EntropyEncoding.CABAC)
                                .slices(1)
                                .unregisteredSeiTimecode(
                                                H264UnregisteredSeiTimecode.DISABLED)
```

```
                    .repeatPps(H264RepeatPps.DISABLED)

                    .adaptiveQuantization(

                                    H264AdaptiveQuantization.HIGH)

                    .spatialAdaptiveQuantization(

                                    H264SpatialAdaptiveQuantization.ENABLED)

                    .temporalAdaptiveQuantization(

                                    H264TemporalAdaptiveQuantization.ENABLED)

                    .flickerAdaptiveQuantization(

                                    H264FlickerAdaptiveQuantization.DISABLED)

                    .softness(0)

                    .interlaceMode(H264InterlaceMode.PROGRESSIVE)

                    .build())

   .build())
                                                                     .build())

.audioDescriptions(AudioDescription.builder()

.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

.languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)

.codecSettings(AudioCodecSettings.builder()

.codec(AudioCodec.AAC).aacSettings(AacSettings

   .builder()

   .codecProfile(AacCodecProfile.LC)

   .rateControlMode(
```

```
                     AacRateControlMode.CBR)

     .codingMode(AacCodingMode.CODING_MODE_2_0)

     .sampleRate(44100)

     .bitrate(96000)

     .rawFormat(AacRawFormat.NONE)

     .specification(AacSpecification.MPEG4)

     .audioDescriptionBroadcasterMix(

                     AacAudioDescriptionBroadcasterMix.NORMAL)

     .build())
                                                                    .build())
                                                    .build())
                                    .build();
                } catch (MediaConvertException e) {
                        e.printStackTrace();
                        System.exit(0);
                }
                return output;
        }
}
```

- For API details, see [CreateJob](#) in *AWS SDK for Java 2.x API Reference*.

GetJob

The following code example shows how to use `GetJob`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetJob {

    public static void main(String[] args) {

        final String usage = "\n" +
                "  <jobId> \n\n" +
                "Where:\n" +
                "  jobId - The job id value.\n\n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String jobId = args[0];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        getSpecificJob(mc, jobId);
        mc.close();
    }

    public static void getSpecificJob(MediaConvertClient mc, String jobId) {
        try {
```

```
            DescribeEndpointsResponse res =
 mc.describeEndpoints(DescribeEndpointsRequest.builder()
                    .maxResults(20)
                    .build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service endpoint
 URL!");
                System.exit(1);
            }
            String endpointURL = res.endpoints().get(0).url();
            MediaConvertClient emc = MediaConvertClient.builder()
                    .region(Region.US_WEST_2)
                    .endpointOverride(URI.create(endpointURL))
                    .build();

            GetJobRequest jobRequest = GetJobRequest.builder()
                    .id(jobId)
                    .build();

            GetJobResponse response = emc.getJob(jobRequest);
            System.out.println("The ARN of the job is " + response.job().arn());

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- For API details, see GetJob in *AWS SDK for Java 2.x API Reference*.

ListJobs

The following code example shows how to use `ListJobs`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/mediaconvert#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListJobs {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            DescribeEndpointsResponse res =
 mc.describeEndpoints(DescribeEndpointsRequest.builder()
```

```
                    .maxResults(20)
                    .build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service endpoint
 URL!");
                System.exit(1);
            }

            String endpointURL = res.endpoints().get(0).url();
            MediaConvertClient emc = MediaConvertClient.builder()
                    .region(Region.US_WEST_2)
                    .endpointOverride(URI.create(endpointURL))
                    .build();

            ListJobsRequest jobsRequest = ListJobsRequest.builder()
                    .maxResults(10)
                    .status("COMPLETE")
                    .build();

            ListJobsResponse jobsResponse = emc.listJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("The JOB ARN is : " + job.arn());
            }

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- For API details, see ListJobs in *AWS SDK for Java 2.x API Reference*.

Migration Hub examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Migration Hub.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

`DeleteProgressUpdateStream`

The following code example shows how to use `DeleteProgressUpdateStream`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DeleteProgressUpdateStreamRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class DeleteProgressStream {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <progressStream>\s

                Where:
                    progressStream - the name of a progress stream to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String progressStream = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        deleteStream(migrationClient, progressStream);
        migrationClient.close();
    }

    public static void deleteStream(MigrationHubClient migrationClient, String
 streamName) {
        try {
            DeleteProgressUpdateStreamRequest deleteProgressUpdateStreamRequest =
 DeleteProgressUpdateStreamRequest
                    .builder()
                    .progressUpdateStreamName(streamName)
                    .build();


 migrationClient.deleteProgressUpdateStream(deleteProgressUpdateStreamRequest);
            System.out.println(streamName + " is deleted");

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
}
```

- For API details, see [DeleteProgressUpdateStream](#) in *AWS SDK for Java 2.x API Reference*.

DescribeApplicationState

The following code example shows how to use `DescribeApplicationState`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateRequest;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAppState {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    DescribeAppState <appId>\s

                Where:
```

```
                    appId -  the application id value.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        describeApplicationState(migrationClient, appId);
        migrationClient.close();
    }

    public static void describeApplicationState(MigrationHubClient migrationClient,
 String appId) {
        try {
            DescribeApplicationStateRequest applicationStateRequest =
 DescribeApplicationStateRequest.builder()
                    .applicationId(appId)
                    .build();

            DescribeApplicationStateResponse applicationStateResponse =
 migrationClient
                    .describeApplicationState(applicationStateRequest);
            System.out.println("The application status is " +
 applicationStateResponse.applicationStatusAsString());

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DescribeApplicationState in *AWS SDK for Java 2.x API Reference*.

DescribeMigrationTask

The following code example shows how to use `DescribeMigrationTask`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/migrationhub#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskRequest;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeMigrationTask {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    DescribeMigrationTask <migrationTask> <progressStream>\s

                Where:
                    migrationTask - the name of a migration task.\s
                    progressStream - the name of a progress stream.\s
                """;

        if (args.length < 2) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        describeMigTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void describeMigTask(MigrationHubClient migrationClient, String
 migrationTask,
            String progressStream) {
        try {
            DescribeMigrationTaskRequest migrationTaskRequestRequest =
 DescribeMigrationTaskRequest.builder()
                    .progressUpdateStream(progressStream)
                    .migrationTaskName(migrationTask)
                    .build();

            DescribeMigrationTaskResponse migrationTaskResponse = migrationClient
                    .describeMigrationTask(migrationTaskRequestRequest);
            System.out.println("The name is " +
 migrationTaskResponse.migrationTask().migrationTaskName());

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DescribeMigrationTask in *AWS SDK for Java 2.x API Reference*.

`ImportMigrationTask`

The following code example shows how to use `ImportMigrationTask`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.CreateProgressUpdateStreamRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ImportMigrationTaskRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportMigrationTask {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <migrationTask> <progressStream>\s

                Where:
                    migrationTask - the name of a migration task.\s
                    progressStream - the name of a progress stream.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String migrationTask = args[0];
```

```
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        importMigrTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void importMigrTask(MigrationHubClient migrationClient, String
 migrationTask, String progressStream) {
        try {
            CreateProgressUpdateStreamRequest progressUpdateStreamRequest =
 CreateProgressUpdateStreamRequest.builder()
                    .progressUpdateStreamName(progressStream)
                    .dryRun(false)
                    .build();

            migrationClient.createProgressUpdateStream(progressUpdateStreamRequest);
            ImportMigrationTaskRequest migrationTaskRequest =
 ImportMigrationTaskRequest.builder()
                    .migrationTaskName(migrationTask)
                    .progressUpdateStream(progressStream)
                    .dryRun(false)
                    .build();

            migrationClient.importMigrationTask(migrationTaskRequest);

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ImportMigrationTask in *AWS SDK for Java 2.x API Reference*.

ListApplications

The following code example shows how to use `ListApplications`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ApplicationState;
import
 software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListApplications {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listApps(migrationClient);
        migrationClient.close();
    }

    public static void listApps(MigrationHubClient migrationClient) {
        try {
            ListApplicationStatesRequest applicationStatesRequest =
 ListApplicationStatesRequest.builder()
                    .maxResults(10)
```

```
                    .build();

            ListApplicationStatesResponse response =
 migrationClient.listApplicationStates(applicationStatesRequest);
            List<ApplicationState> apps = response.applicationStateList();
            for (ApplicationState appState : apps) {
                System.out.println("App Id is " + appState.applicationId());
                System.out.println("The status is " +
 appState.applicationStatus().toString());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [ListApplications](#) in *AWS SDK for Java 2.x API Reference*.

ListCreatedArtifacts

The following code example shows how to use `ListCreatedArtifacts`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;
```

```
/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCreatedArtifacts {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listArtifacts(migrationClient);
        migrationClient.close();
    }

    public static void listArtifacts(MigrationHubClient migrationClient) {
        try {
            ListCreatedArtifactsRequest listCreatedArtifactsRequest =
 ListCreatedArtifactsRequest.builder()
                    .maxResults(10)
                    .migrationTaskName("SampleApp5")
                    .progressUpdateStream("ProgressSteamB")
                    .build();

            ListCreatedArtifactsResponse response =
 migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
            List<CreatedArtifact> apps = response.createdArtifactList();
            for (CreatedArtifact artifact : apps) {
                System.out.println("APp Id is " + artifact.description());
                System.out.println("The name is " + artifact.name());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListCreatedArtifacts in *AWS SDK for Java 2.x API Reference*.

ListMigrationTasks

The following code example shows how to use `ListMigrationTasks`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationTaskSummary;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMigrationTasks {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listMigrTasks(migrationClient);
```

```
        migrationClient.close();
    }

    public static void listMigrTasks(MigrationHubClient migrationClient) {
        try {
            ListMigrationTasksRequest listMigrationTasksRequest =
 ListMigrationTasksRequest.builder()
                    .maxResults(10)
                    .build();

            ListMigrationTasksResponse response =
 migrationClient.listMigrationTasks(listMigrationTasksRequest);
            List<MigrationTaskSummary> migrationList =
 response.migrationTaskSummaryList();
            for (MigrationTaskSummary migration : migrationList) {
                System.out.println("Migration task name is " +
 migration.migrationTaskName());
                System.out.println("The Progress update stream is " +
 migration.progressUpdateStream());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListMigrationTasks in *AWS SDK for Java 2.x API Reference*.

Amazon Personalize examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

Actions

CreateBatchInferenceJob

The following code example shows how to use `CreateBatchInferenceJob`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        public static String createPersonalizeBatchInferenceJob(PersonalizeClient
 personalizeClient,
                        String solutionVersionArn,
                        String jobName,
                        String s3InputDataSourcePath,
                        String s3DataDestinationPath,
                        String roleArn,
                        String explorationWeight,
                        String explorationItemAgeCutOff) {

                long waitInMilliseconds = 60 * 1000;
                String status;
                String batchInferenceJobArn;

                try {

                        // Set up data input and output parameters.
                        S3DataConfig inputSource = S3DataConfig.builder()
                                        .path(s3InputDataSourcePath)
                                        .build();
```

```
                        S3DataConfig outputDestination = S3DataConfig.builder()
                                        .path(s3DataDestinationPath)
                                        .build();

                        BatchInferenceJobInput jobInput =
BatchInferenceJobInput.builder()
                                        .s3DataSource(inputSource)
                                        .build();

                        BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
                                        .s3DataDestination(outputDestination)
                                        .build();

                        // Optional code to build the User-Personalization specific
item exploration
                        // config.
                        HashMap<String, String> explorationConfig = new HashMap<>();

                        explorationConfig.put("explorationWeight",
explorationWeight);
                        explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

                        BatchInferenceJobConfig jobConfig =
BatchInferenceJobConfig.builder()
                                        .itemExplorationConfig(explorationConfig)
                                        .build();

                        // End optional User-Personalization recipe specific code.

                        CreateBatchInferenceJobRequest
createBatchInferenceJobRequest = CreateBatchInferenceJobRequest
                                        .builder()
                                        .solutionVersionArn(solutionVersionArn)
                                        .jobInput(jobInput)
                                        .jobOutput(jobOutputLocation)
                                        .jobName(jobName)
                                        .roleArn(roleArn)
                                        .batchInferenceJobConfig(jobConfig) //
Optional
                                        .build();
```

```
                        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
                                        .batchInferenceJobArn();

                        DescribeBatchInferenceJobRequest
describeBatchInferenceJobRequest = DescribeBatchInferenceJobRequest
                                        .builder()
                                        .batchInferenceJobArn(batchInferenceJobArn)
                                        .build();

                        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
                        while (Instant.now().getEpochSecond() < maxTime) {

                                BatchInferenceJob batchInferenceJob =
personalizeClient

.describeBatchInferenceJob(describeBatchInferenceJobRequest)
                                                .batchInferenceJob();

                                status = batchInferenceJob.status();
                                System.out.println("Batch inference job status: " +
status);

                                if (status.equals("ACTIVE") || status.equals("CREATE
FAILED")) {
                                        break;
                                }
                                try {
                                        Thread.sleep(waitInMilliseconds);
                                } catch (InterruptedException e) {
                                        System.out.println(e.getMessage());
                                }
                        }
                        return batchInferenceJobArn;

                } catch (PersonalizeException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return "";
        }
```

- For API details, see [CreateBatchInferenceJob](#) in *AWS SDK for Java 2.x API Reference*.

CreateCampaign

The following code example shows how to use CreateCampaign.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples).

```
    public static void createPersonalCompaign(PersonalizeClient personalizeClient,
 String solutionVersionArn,
            String name) {

        try {
            CreateCampaignRequest createCampaignRequest =
 CreateCampaignRequest.builder()
                    .minProvisionedTPS(1)
                    .solutionVersionArn(solutionVersionArn)
                    .name(name)
                    .build();

            CreateCampaignResponse campaignResponse =
 personalizeClient.createCampaign(createCampaignRequest);
            System.out.println("The campaign ARN is " +
 campaignResponse.campaignArn());

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [CreateCampaign]() in *AWS SDK for Java 2.x API Reference*.

CreateDataset

The following code example shows how to use CreateDataset.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static String createDataset(PersonalizeClient personalizeClient,
        String datasetName,
        String datasetGroupArn,
        String datasetType,
        String schemaArn) {
    try {
        CreateDatasetRequest request = CreateDatasetRequest.builder()
                .name(datasetName)
                .datasetGroupArn(datasetGroupArn)
                .datasetType(datasetType)
                .schemaArn(schemaArn)
                .build();

        String datasetArn = personalizeClient.createDataset(request)
                .datasetArn();
        System.out.println("Dataset " + datasetName + " created.");
        return datasetArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see CreateDataset in *AWS SDK for Java 2.x API Reference*.

`CreateDatasetExportJob`

The following code example shows how to use `CreateDatasetExportJob`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createDatasetExportJob(PersonalizeClient personalizeClient,
            String jobName,
            String datasetArn,
            IngestionMode ingestionMode,
            String roleArn,
            String s3BucketPath,
            String kmsKeyArn) {

        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String status = null;

        try {

            S3DataConfig exportS3DataConfig =
S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
            DatasetExportJobOutput jobOutput =
DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig)
                    .build();

            CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
                    .jobName(jobName)
                    .datasetArn(datasetArn)
                    .ingestionMode(ingestionMode)
                    .jobOutput(jobOutput)
                    .roleArn(roleArn)
                    .build();

            String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

            DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
                    .datasetExportJobArn(datasetExportJobArn)
```

```
                    .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetExportJob datasetExportJob = personalizeClient
                        .describeDatasetExportJob(describeDatasetExportJobRequest)
                        .datasetExportJob();

                status = datasetExportJob.status();
                System.out.println("Export job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    return status;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- For API details, see [CreateDatasetExportJob](#) in *AWS SDK for Java 2.x API Reference*.

`CreateDatasetGroup`

The following code example shows how to use `CreateDatasetGroup`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static String createDatasetGroup(PersonalizeClient personalizeClient,
 String datasetGroupName) {

        try {
            CreateDatasetGroupRequest createDatasetGroupRequest =
 CreateDatasetGroupRequest.builder()
                    .name(datasetGroupName)
                    .build();
            return
 personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

Create a domain dataset group.

```
    public static String createDomainDatasetGroup(PersonalizeClient
 personalizeClient,
            String datasetGroupName,
            String domain) {

        try {
            CreateDatasetGroupRequest createDatasetGroupRequest =
 CreateDatasetGroupRequest.builder()
                    .name(datasetGroupName)
                    .domain(domain)
                    .build();
            return
 personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- For API details, see CreateDatasetGroup in *AWS SDK for Java 2.x API Reference*.

`CreateDatasetImportJob`

The following code example shows how to use `CreateDatasetImportJob`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/personalize#code-examples).

```
    public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
            String jobName,
            String datasetArn,
            String s3BucketPath,
            String roleArn) {

        long waitInMilliseconds = 60 * 1000;
        String status;
        String datasetImportJobArn;

        try {
            DataSource importDataSource = DataSource.builder()
                    .dataLocation(s3BucketPath)
                    .build();

            CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
                    .datasetArn(datasetArn)
                    .dataSource(importDataSource)
                    .jobName(jobName)
                    .roleArn(roleArn)
                    .build();

            datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
                    .datasetImportJobArn();
            DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
                    .datasetImportJobArn(datasetImportJobArn)
                    .build();
```

```
            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetImportJob datasetImportJob = personalizeClient
                        .describeDatasetImportJob(describeDatasetImportJobRequest)
                        .datasetImportJob();

                status = datasetImportJob.status();
                System.out.println("Dataset import job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return datasetImportJobArn;

        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- For API details, see CreateDatasetImportJob in *AWS SDK for Java 2.x API Reference*.

`CreateEventTracker`

The following code example shows how to use `CreateEventTracker`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createEventTracker(PersonalizeClient personalizeClient,
String eventTrackerName,
            String datasetGroupArn) {

        String eventTrackerId = "";
        String eventTrackerArn;
        long maxTime = 3 * 60 * 60; // 3 hours
        long waitInMilliseconds = 20 * 1000; // 20 seconds
        String status;

        try {

            CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
                    .name(eventTrackerName)
                    .datasetGroupArn(datasetGroupArn)
                    .build();

            CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient
                    .createEventTracker(createEventTrackerRequest);

            eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
            eventTrackerId = createEventTrackerResponse.trackingId();
            System.out.println("Event tracker ARN: " + eventTrackerArn);
            System.out.println("Event tracker ID: " + eventTrackerId);

            maxTime = Instant.now().getEpochSecond() + maxTime;

            DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
                    .eventTrackerArn(eventTrackerArn)
                    .build();

            while (Instant.now().getEpochSecond() < maxTime) {

                status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
                System.out.println("EventTracker status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
```

```
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return eventTrackerId;
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return eventTrackerId;
    }
```

- For API details, see [CreateEventTracker](#) in *AWS SDK for Java 2.x API Reference*.

CreateFilter

The following code example shows how to use `CreateFilter`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static String createFilter(PersonalizeClient personalizeClient,
            String filterName,
            String datasetGroupArn,
            String filterExpression) {
        try {
            CreateFilterRequest request = CreateFilterRequest.builder()
                    .name(filterName)
                    .datasetGroupArn(datasetGroupArn)
                    .filterExpression(filterExpression)
                    .build();

            return personalizeClient.createFilter(request).filterArn();
```

```
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateFilter in *AWS SDK for Java 2.x API Reference*.

`CreateRecommender`

The following code example shows how to use `CreateRecommender`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createRecommender(PersonalizeClient personalizeClient,
            String name,
            String datasetGroupArn,
            String recipeArn) {

        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String recommenderStatus = "";

        try {
            CreateRecommenderRequest createRecommenderRequest =
 CreateRecommenderRequest.builder()
                    .datasetGroupArn(datasetGroupArn)
                    .name(name)
                    .recipeArn(recipeArn)
                    .build();

            CreateRecommenderResponse recommenderResponse = personalizeClient
                    .createRecommender(createRecommenderRequest);
```

```
            String recommenderArn = recommenderResponse.recommenderArn();
            System.out.println("The recommender ARN is " + recommenderArn);

            DescribeRecommenderRequest describeRecommenderRequest =
 DescribeRecommenderRequest.builder()
                    .recommenderArn(recommenderArn)
                    .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                recommenderStatus =
 personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
                        .status();
                System.out.println("Recommender status: " + recommenderStatus);

                if (recommenderStatus.equals("ACTIVE") ||
 recommenderStatus.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return recommenderArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateRecommender in *AWS SDK for Java 2.x API Reference*.

`CreateSchema`

The following code example shows how to use `CreateSchema`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createSchema(PersonalizeClient personalizeClient, String
 schemaName, String filePath) {

        String schema = null;
        try {
            schema = new String(Files.readAllBytes(Paths.get(filePath)));
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }

        try {
            CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
                    .name(schemaName)
                    .schema(schema)
                    .build();

            String schemaArn =
 personalizeClient.createSchema(createSchemaRequest).schemaArn();

            System.out.println("Schema arn: " + schemaArn);

            return schemaArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

Create a schema with a domain.

```
    public static String createDomainSchema(PersonalizeClient personalizeClient,
 String schemaName, String domain,
            String filePath) {

        String schema = null;
        try {
            schema = new String(Files.readAllBytes(Paths.get(filePath)));
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }

        try {
            CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
                    .name(schemaName)
                    .domain(domain)
                    .schema(schema)
                    .build();

            String schemaArn =
 personalizeClient.createSchema(createSchemaRequest).schemaArn();

            System.out.println("Schema arn: " + schemaArn);

            return schemaArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateSchema in *AWS SDK for Java 2.x API Reference*.

CreateSolution

The following code example shows how to use `CreateSolution`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createPersonalizeSolution(PersonalizeClient
 personalizeClient,
            String datasetGroupArn,
            String solutionName,
            String recipeArn) {

        try {
            CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
                    .name(solutionName)
                    .datasetGroupArn(datasetGroupArn)
                    .recipeArn(recipeArn)
                    .build();

            CreateSolutionResponse solutionResponse =
 personalizeClient.createSolution(solutionRequest);
            return solutionResponse.solutionArn();

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateSolution in *AWS SDK for Java 2.x API Reference*.

`CreateSolutionVersion`

The following code example shows how to use `CreateSolutionVersion`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/personalize#code-examples).

```
    public static String createPersonalizeSolutionVersion(PersonalizeClient
 personalizeClient, String solutionArn) {
        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String solutionStatus = "";
        String solutionVersionStatus = "";
        String solutionVersionArn = "";

        try {
            DescribeSolutionRequest describeSolutionRequest =
 DescribeSolutionRequest.builder()
                    .solutionArn(solutionArn)
                    .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            // Wait until solution is active.
            while (Instant.now().getEpochSecond() < maxTime) {

                solutionStatus =
 personalizeClient.describeSolution(describeSolutionRequest).solution().status();
                System.out.println("Solution status: " + solutionStatus);

                if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
 FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
```

```
            if (solutionStatus.equals("ACTIVE")) {

                CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
                        .solutionArn(solutionArn)
                        .build();

                CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
                        .createSolutionVersion(createSolutionVersionRequest);
                solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

                System.out.println("Solution version ARN: " + solutionVersionArn);

                DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
                        .solutionVersionArn(solutionVersionArn)
                        .build();

                while (Instant.now().getEpochSecond() < maxTime) {

                    solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                            .solutionVersion().status();
                    System.out.println("Solution version status: " +
solutionVersionStatus);

                    if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                        break;
                    }
                    try {
                        Thread.sleep(waitInMilliseconds);
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                return solutionVersionArn;
            }
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return "";
    }
```

- For API details, see CreateSolutionVersion in *AWS SDK for Java 2.x API Reference*.

DeleteCampaign

The following code example shows how to use `DeleteCampaign`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
 String campaignArn) {

        try {
            DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();

            personalizeClient.deleteCampaign(campaignRequest);

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteCampaign in *AWS SDK for Java 2.x API Reference*.

DeleteEventTracker

The following code example shows how to use `DeleteEventTracker`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteEventTracker(PersonalizeClient personalizeClient,
 String eventTrackerArn) {
        try {
            DeleteEventTrackerRequest deleteEventTrackerRequest =
 DeleteEventTrackerRequest.builder()
                    .eventTrackerArn(eventTrackerArn)
                    .build();

            int status =
 personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCod

            System.out.println("Status code:" + status);

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteEventTracker in *AWS SDK for Java 2.x API Reference*.

`DeleteSolution`

The following code example shows how to use `DeleteSolution`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteGivenSolution(PersonalizeClient personalizeClient,
 String solutionArn) {

        try {
            DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
                    .solutionArn(solutionArn)
                    .build();

            personalizeClient.deleteSolution(solutionRequest);
            System.out.println("Done");

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteSolution in *AWS SDK for Java 2.x API Reference*.

DescribeCampaign

The following code example shows how to use `DescribeCampaign`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
 String campaignArn) {

        try {
            DescribeCampaignRequest campaignRequest =
 DescribeCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();
```

```
            DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
            Campaign myCampaign = campaignResponse.campaign();
            System.out.println("The Campaign name is " + myCampaign.name());
            System.out.println("The Campaign status is " + myCampaign.status());

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DescribeCampaign](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeRecipe`

The following code example shows how to use `DescribeRecipe`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {

        try {
            DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
                    .recipeArn(recipeArn)
                    .build();

            DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
            System.out.println("The recipe name is " +
recipeResponse.recipe().name());

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
```

- For API details, see DescribeRecipe in *AWS SDK for Java 2.x API Reference*.

DescribeSolution

The following code example shows how to use `DescribeSolution`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeSpecificSolution(PersonalizeClient personalizeClient,
 String solutionArn) {

        try {
            DescribeSolutionRequest solutionRequest =
 DescribeSolutionRequest.builder()
                    .solutionArn(solutionArn)
                    .build();

            DescribeSolutionResponse response =
 personalizeClient.describeSolution(solutionRequest);
            System.out.println("The Solution name is " +
 response.solution().name());

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeSolution in *AWS SDK for Java 2.x API Reference*.

ListCampaigns

The following code example shows how to use `ListCampaigns`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/personalize#code-examples).

```
    public static void listAllCampaigns(PersonalizeClient personalizeClient, String
 solutionArn) {

        try {
            ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
                    .maxResults(10)
                    .solutionArn(solutionArn)
                    .build();

            ListCampaignsResponse response =
 personalizeClient.listCampaigns(campaignsRequest);
            List<CampaignSummary> campaigns = response.campaigns();
            for (CampaignSummary campaign : campaigns) {
                System.out.println("Campaign name is : " + campaign.name());
                System.out.println("Campaign ARN is : " + campaign.campaignArn());
            }

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [ListCampaigns](https://docs.aws.amazon.com/goto/SdkForJavaV2/personalize-2018-05-22/ListCampaigns) in *AWS SDK for Java 2.x API Reference*.

ListDatasetGroups

The following code example shows how to use `ListDatasetGroups`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/personalize#code-examples).

```
    public static void listDSGroups(PersonalizeClient personalizeClient) {

        try {
            ListDatasetGroupsRequest groupsRequest =
 ListDatasetGroupsRequest.builder()
                    .maxResults(15)
                    .build();

            ListDatasetGroupsResponse groupsResponse =
 personalizeClient.listDatasetGroups(groupsRequest);
            List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
            for (DatasetGroupSummary group : groups) {
                System.out.println("The DataSet name is : " + group.name());
                System.out.println("The DataSet ARN is : " +
 group.datasetGroupArn());
            }

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListDatasetGroups in *AWS SDK for Java 2.x API Reference*.

`ListRecipes`

The following code example shows how to use `ListRecipes`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllRecipes(PersonalizeClient personalizeClient) {

        try {
            ListRecipesRequest recipesRequest = ListRecipesRequest.builder()
                    .maxResults(15)
                    .build();

            ListRecipesResponse response =
 personalizeClient.listRecipes(recipesRequest);
            List<RecipeSummary> recipes = response.recipes();
            for (RecipeSummary recipe : recipes) {
                System.out.println("The recipe ARN is: " + recipe.recipeArn());
                System.out.println("The recipe name is: " + recipe.name());
            }

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListRecipes in *AWS SDK for Java 2.x API Reference*.

`ListSolutions`

The following code example shows how to use `ListSolutions`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllSolutions(PersonalizeClient personalizeClient, String 
datasetGroupArn) {

        try {
            ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()
                    .maxResults(10)
                    .datasetGroupArn(datasetGroupArn)
                    .build();

            ListSolutionsResponse response = 
personalizeClient.listSolutions(solutionsRequest);
            List<SolutionSummary> solutions = response.solutions();
            for (SolutionSummary solution : solutions) {
                System.out.println("The solution ARN is: " + 
solution.solutionArn());
                System.out.println("The solution name is: " + solution.name());
            }

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListSolutions in *AWS SDK for Java 2.x API Reference*.

`UpdateCampaign`

The following code example shows how to use `UpdateCampaign`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/personalize#code-examples).

```
    public static String updateCampaign(PersonalizeClient personalizeClient,
            String campaignArn,
            String solutionVersionArn,
            Integer minProvisionedTPS) {

        try {
            // build the updateCampaignRequest
            UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .solutionVersionArn(solutionVersionArn)
                    .minProvisionedTPS(minProvisionedTPS)
                    .build();

            // update the campaign
            personalizeClient.updateCampaign(updateCampaignRequest);

            DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();

            DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
            Campaign updatedCampaign = campaignResponse.campaign();

            System.out.println("The Campaign status is " +
updatedCampaign.status());
            return updatedCampaign.status();

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return "";
    }
```

- For API details, see UpdateCampaign in *AWS SDK for Java 2.x API Reference*.

Amazon Personalize Events examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize Events.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

`PutEvents`

The following code example shows how to use `PutEvents`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static int putItems(PersonalizeEventsClient personalizeEventsClient,
                               String datasetArn,
```

```
                         String item1Id,
                         String item1PropertyName,
                         String item1PropertyValue,
                         String item2Id,
                         String item2PropertyName,
                         String item2PropertyValue) {

                 int responseCode = 0;
                 ArrayList<Item> items = new ArrayList<>();

                 try {
                         Item item1 = Item.builder()
                                         .itemId(item1Id)
                                         .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                         item1PropertyName,
 item1PropertyValue))
                                         .build();

                         items.add(item1);

                         Item item2 = Item.builder()
                                         .itemId(item2Id)
                                         .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                         item2PropertyName,
 item2PropertyValue))
                                         .build();

                         items.add(item2);

                         PutItemsRequest putItemsRequest = PutItemsRequest.builder()
                                         .datasetArn(datasetArn)
                                         .items(items)
                                         .build();

                         responseCode =
 personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
                         System.out.println("Response code: " + responseCode);
                         return responseCode;

                 } catch (PersonalizeEventsException e) {
                         System.out.println(e.awsErrorDetails().errorMessage());
                 }
```

```
                return responseCode;
        }
```

- For API details, see PutEvents in *AWS SDK for Java 2.x API Reference*.

PutUsers

The following code example shows how to use `PutUsers`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
                        String datasetArn,
                        String user1Id,
                        String user1PropertyName,
                        String user1PropertyValue,
                        String user2Id,
                        String user2PropertyName,
                        String user2PropertyValue) {

                int responseCode = 0;
                ArrayList<User> users = new ArrayList<>();

                try {
                        User user1 = User.builder()
                                        .userId(user1Id)
                                        .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                        user1PropertyName,
 user1PropertyValue))
                                        .build();

                        users.add(user1);

                        User user2 = User.builder()
```

```
                                        .userId(user2Id)
                                        .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                        user2PropertyName,
 user2PropertyValue))
                                        .build();

                        users.add(user2);

                        PutUsersRequest putUsersRequest = PutUsersRequest.builder()
                                        .datasetArn(datasetArn)
                                        .users(users)
                                        .build();

                        responseCode =
 personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
                        System.out.println("Response code: " + responseCode);
                        return responseCode;

                } catch (PersonalizeEventsException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return responseCode;
        }
```

- For API details, see PutUsers in *AWS SDK for Java 2.x API Reference*.

Amazon Personalize Runtime examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize Runtime.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

Actions

GetPersonalizedRanking

The following code example shows how to use `GetPersonalizedRanking`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
 personalizeRuntimeClient,
            String campaignArn,
            String userId,
            ArrayList<String> items) {

        try {
            GetPersonalizedRankingRequest rankingRecommendationsRequest =
 GetPersonalizedRankingRequest.builder()
                    .campaignArn(campaignArn)
                    .userId(userId)
                    .inputList(items)
                    .build();

            GetPersonalizedRankingResponse recommendationsResponse =
 personalizeRuntimeClient
                    .getPersonalizedRanking(rankingRecommendationsRequest);
            List<PredictedItem> rankedItems =
 recommendationsResponse.personalizedRanking();
            int rank = 1;
            for (PredictedItem item : rankedItems) {
                System.out.println("Item ranked at position " + rank + " details");
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());
```

```
                System.out.println("----------------------------------------------");
                rank++;
            }
            return rankedItems;
        } catch (PersonalizeRuntimeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- For API details, see GetPersonalizedRanking in *AWS SDK for Java 2.x API Reference*.

GetRecommendations

The following code example shows how to use GetRecommendations.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Get a list of recommended items.

```
    public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
 String campaignArn, String userId) {

        try {
            GetRecommendationsRequest recommendationsRequest =
 GetRecommendationsRequest.builder()
                    .campaignArn(campaignArn)
                    .numResults(20)
                    .userId(userId)
                    .build();

            GetRecommendationsResponse recommendationsResponse =
 personalizeRuntimeClient
```

```
                    .getRecommendations(recommendationsRequest);
            List<PredictedItem> items = recommendationsResponse.itemList();
            for (PredictedItem item : items) {
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());
            }

        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Get a list of recommended items from a recommender created in a domain dataset group.

```
    public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
 String recommenderArn,
            String userId) {

        try {
            GetRecommendationsRequest recommendationsRequest =
 GetRecommendationsRequest.builder()
                    .recommenderArn(recommenderArn)
                    .numResults(20)
                    .userId(userId)
                    .build();

            GetRecommendationsResponse recommendationsResponse =
 personalizeRuntimeClient
                    .getRecommendations(recommendationsRequest);
            List<PredictedItem> items = recommendationsResponse.itemList();

            for (PredictedItem item : items) {
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());
            }
        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Use a filter when requesting recommendations.

```
    public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
            String campaignArn,
            String userId,
            String filterArn,
            String parameter1Name,
            String parameter1Value1,
            String parameter1Value2,
            String parameter2Name,
            String parameter2Value) {

        try {

            Map<String, String> filterValues = new HashMap<>();

            filterValues.put(parameter1Name, String.format("\"%1$s\",\"%2$s\"",
                    parameter1Value1, parameter1Value2));
            filterValues.put(parameter2Name, String.format("\"%1$s\"",
                    parameter2Value));

            GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                    .campaignArn(campaignArn)
                    .numResults(20)
                    .userId(userId)
                    .filterArn(filterArn)
                    .filterValues(filterValues)
                    .build();

            GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                    .getRecommendations(recommendationsRequest);
            List<PredictedItem> items = recommendationsResponse.itemList();

            for (PredictedItem item : items) {
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());
            }
        } catch (PersonalizeRuntimeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
```

- For API details, see GetRecommendations in *AWS SDK for Java 2.x API Reference*.

Amazon Pinpoint examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Pinpoint.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

`CreateApp`

The following code example shows how to use `CreateApp`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateAppRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.CreateAppResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateApplicationRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateApp {
    public static void main(String[] args) {
        final String usage = """

                 Usage:  <appName>

                 Where:
                  appName - The name of the application to create.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String appName = args[0];
        System.out.println("Creating an application with name: " + appName);

        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String appID = createApplication(pinpoint, appName);
        System.out.println("App ID is: " + appID);
        pinpoint.close();
    }

    public static String createApplication(PinpointClient pinpoint, String appName)
 {
        try {
            CreateApplicationRequest appRequest = CreateApplicationRequest.builder()
                    .name(appName)
```

```
                    .build();

            CreateAppRequest request = CreateAppRequest.builder()
                    .createApplicationRequest(appRequest)
                    .build();

            CreateAppResponse result = pinpoint.createApp(request);
            return result.applicationResponse().id();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see CreateApp in *AWS SDK for Java 2.x API Reference*.

`CreateCampaign`

The following code example shows how to use `CreateCampaign`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a campaign.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.Message;
import software.amazon.awssdk.services.pinpoint.model.Schedule;
import software.amazon.awssdk.services.pinpoint.model.Action;
import software.amazon.awssdk.services.pinpoint.model.MessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.WriteCampaignRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCampaign {
    public static void main(String[] args) {

        final String usage = """

                Usage:   <appId> <segmentId>

                Where:
                  appId - The ID of the application to create the campaign in.
                  segmentId - The ID of the segment to create the campaign from.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String segmentId = args[1];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPinCampaign(pinpoint, appId, segmentId);
        pinpoint.close();
    }

    public static void createPinCampaign(PinpointClient pinpoint, String appId,
 String segmentId) {
        CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
        System.out.println("Campaign " + result.name() + " created.");
        System.out.println(result.description());
```

```
    }

    public static CampaignResponse createCampaign(PinpointClient client, String
 appID, String segmentID) {

        try {
            Schedule schedule = Schedule.builder()
                    .startTime("IMMEDIATE")
                    .build();

            Message defaultMessage = Message.builder()
                    .action(Action.OPEN_APP)
                    .body("My message body.")
                    .title("My message title.")
                    .build();

            MessageConfiguration messageConfiguration =
 MessageConfiguration.builder()
                    .defaultMessage(defaultMessage)
                    .build();

            WriteCampaignRequest request = WriteCampaignRequest.builder()
                    .description("My description")
                    .schedule(schedule)
                    .name("MyCampaign")
                    .segmentId(segmentID)
                    .messageConfiguration(messageConfiguration)
                    .build();

            CreateCampaignResponse result =
 client.createCampaign(CreateCampaignRequest.builder()
                    .applicationId(appID)
                    .writeCampaignRequest(request).build());

            System.out.println("Campaign ID: " + result.campaignResponse().id());
            return result.campaignResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

```
}
```

- For API details, see CreateCampaign in *AWS SDK for Java 2.x API Reference*.

CreateExportJob

The following code example shows how to use `CreateExportJob`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Export an endpoint.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.ExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
```

```
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

/**
 * To run this code example, you need to create an AWS Identity and Access
 * Management (IAM) role with the correct policy as described in this
 * documentation:
 * https://docs.aws.amazon.com/pinpoint/latest/developerguide/audience-data-
export.html
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ExportEndpoints {
    public static void main(String[] args) {
        final String usage = """

                This program performs the following steps:

                1. Exports the endpoints to an Amazon S3 bucket.
                2. Downloads the exported endpoints files from Amazon S3.
                3. Parses the endpoints files to obtain the endpoint IDs and prints
 them.
                Usage: ExportEndpoints <applicationId> <s3BucketName>
 <iamExportRoleArn> <path>

                Where:
                  applicationId - The ID of the Amazon Pinpoint application that has
 the endpoint.
                  s3BucketName - The name of the Amazon S3 bucket to export the JSON
 file to.\s
                  iamExportRoleArn - The ARN of an IAM role that grants Amazon
 Pinpoint write permissions to the S3 bucket.  path - The path where the files
 downloaded from the Amazon S3 bucket are written (for example, C:/AWS/).
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String applicationId = args[0];
        String s3BucketName = args[1];
        String iamExportRoleArn = args[2];
        String path = args[3];
        System.out.println("Deleting an application with ID: " + applicationId);

        Region region = Region.US_EAST_1;
        PinpointClient pinpoint = PinpointClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        exportAllEndpoints(pinpoint, s3Client, applicationId, s3BucketName, path,
iamExportRoleArn);
        pinpoint.close();
        s3Client.close();
    }

    public static void exportAllEndpoints(PinpointClient pinpoint,
            S3Client s3Client,
            String applicationId,
            String s3BucketName,
            String path,
            String iamExportRoleArn) {

        try {
            List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
s3BucketName, iamExportRoleArn,
                    applicationId);
            List<String> endpointFileKeys = objectKeys.stream().filter(o ->
o.endsWith(".gz"))
                    .collect(Collectors.toList());
            downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
    public static List<String> exportEndpointsToS3(PinpointClient pinpoint, S3Client
 s3Client, String s3BucketName,
            String iamExportRoleArn, String applicationId) {

        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
        String endpointsKeyPrefix = "exports/" + applicationId + "_" +
 dateFormat.format(new Date());
        String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix +
 "/";
        List<String> objectKeys = new ArrayList<>();
        String key;

        try {
            // Defines the export job that Amazon Pinpoint runs.
            ExportJobRequest jobRequest = ExportJobRequest.builder()
                    .roleArn(iamExportRoleArn)
                    .s3UrlPrefix(s3UrlPrefix)
                    .build();

            CreateExportJobRequest exportJobRequest =
 CreateExportJobRequest.builder()
                    .applicationId(applicationId)
                    .exportJobRequest(jobRequest)
                    .build();

            System.out.format("Exporting endpoints from Amazon Pinpoint application
 %s to Amazon S3 " +
                    "bucket %s . . .\n", applicationId, s3BucketName);

            CreateExportJobResponse exportResult =
 pinpoint.createExportJob(exportJobRequest);
            String jobId = exportResult.exportJobResponse().id();
            System.out.println(jobId);
            printExportJobStatus(pinpoint, applicationId, jobId);

            ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
                    .bucket(s3BucketName)
                    .prefix(endpointsKeyPrefix)
                    .build();

            // Create a list of object keys.
            ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
```

```
            List<S3Object> objects = v2Response.contents();
            for (S3Object object : objects) {
                key = object.key();
                objectKeys.add(key);
            }

            return objectKeys;

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void printExportJobStatus(PinpointClient pinpointClient,
            String applicationId,
            String jobId) {

        GetExportJobResponse getExportJobResult;
        String status;

        try {
            // Checks the job status until the job completes or fails.
            GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
                    .jobId(jobId)
                    .applicationId(applicationId)
                    .build();

            do {
                getExportJobResult = pinpointClient.getExportJob(exportJobRequest);
                status =
 getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
                System.out.format("Export job %s . . .\n", status);
                TimeUnit.SECONDS.sleep(3);

            } while (!status.equals("COMPLETED") && !status.equals("FAILED"));

            if (status.equals("COMPLETED")) {
                System.out.println("Finished exporting endpoints.");
            } else {
                System.err.println("Failed to export endpoints.");
                System.exit(1);
            }
```

```
        } catch (PinpointException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Download files from an Amazon S3 bucket and write them to the path location.
    public static void downloadFromS3(S3Client s3Client, String path, String
 s3BucketName, List<String> objectKeys) {

        String newPath;
        try {
            for (String key : objectKeys) {
                GetObjectRequest objectRequest = GetObjectRequest.builder()
                        .bucket(s3BucketName)
                        .key(key)
                        .build();

                ResponseBytes<GetObjectResponse> objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
                byte[] data = objectBytes.asByteArray();

                // Write the data to a local file.
                String fileSuffix = new
 SimpleDateFormat("yyyyMMddHHmmss").format(new Date());
                newPath = path + fileSuffix + ".gz";
                File myFile = new File(newPath);
                OutputStream os = new FileOutputStream(myFile);
                os.write(data);
            }
            System.out.println("Download finished.");

        } catch (S3Exception | NullPointerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see CreateExportJob in *AWS SDK for Java 2.x API Reference*.

`CreateImportJob`

The following code example shows how to use `CreateImportJob`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/pinpoint#code-examples).

Import a segment.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.ImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.ImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.Format;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportSegment {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId> <bucket> <key> <roleArn>\s

                Where:
                  appId - The application ID to create a segment for.
                  bucket - The name of the Amazon S3 bucket that contains the
 segment definitons.
                  key - The key of the S3 object.
```

```
                  roleArn - ARN of the role that allows Amazon Pinpoint to
 access S3. You need to set trust management for this to work. See https://
docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_principal.html
                  """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String bucket = args[1];
        String key = args[2];
        String roleArn = args[3];

        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        ImportJobResponse response = createImportSegment(pinpoint, appId, bucket,
 key, roleArn);
        System.out.println("Import job for " + bucket + " submitted.");
        System.out.println("See application " + response.applicationId() + " for
 import job status.");
        System.out.println("See application " + response.jobStatus() + " for import
 job status.");
        pinpoint.close();
    }

    public static ImportJobResponse createImportSegment(PinpointClient client,
            String appId,
            String bucket,
            String key,
            String roleArn) {

        try {
            ImportJobRequest importRequest = ImportJobRequest.builder()
                    .defineSegment(true)
                    .registerEndpoints(true)
                    .roleArn(roleArn)
                    .format(Format.JSON)
                    .s3Url("s3://" + bucket + "/" + key)
                    .build();
```

```
            CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
                    .importJobRequest(importRequest)
                    .applicationId(appId)
                    .build();

            CreateImportJobResponse jobResponse =
 client.createImportJob(jobRequest);
            return jobResponse.importJobResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- For API details, see CreateImportJob in *AWS SDK for Java 2.x API Reference*.

CreateSegment

The following code example shows how to use `CreateSegment`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AttributeDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.AttributeType;
import software.amazon.awssdk.services.pinpoint.model.RecencyDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentBehaviors;
import software.amazon.awssdk.services.pinpoint.model.SegmentDemographics;
import software.amazon.awssdk.services.pinpoint.model.SegmentLocation;
import software.amazon.awssdk.services.pinpoint.model.SegmentDimensions;
```

```
import software.amazon.awssdk.services.pinpoint.model.WriteSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateSegment {
        public static void main(String[] args) {
                final String usage = """

                                Usage:   <appId>

                                Where:
                                  appId - The application ID to create a segment
 for.

                                """;

                if (args.length != 1) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String appId = args[0];
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                SegmentResponse result = createSegment(pinpoint, appId);
                System.out.println("Segment " + result.name() + " created.");
                System.out.println(result.segmentType());
                pinpoint.close();
        }
```

```
        public static SegmentResponse createSegment(PinpointClient client, String
appId) {
                try {
                        Map<String, AttributeDimension> segmentAttributes = new
HashMap<>();
                        segmentAttributes.put("Team", AttributeDimension.builder()
                                        .attributeType(AttributeType.INCLUSIVE)
                                        .values("Lakers")
                                        .build());

                        RecencyDimension recencyDimension =
RecencyDimension.builder()
                                        .duration("DAY_30")
                                        .recencyType("ACTIVE")
                                        .build();

                        SegmentBehaviors segmentBehaviors =
SegmentBehaviors.builder()
                                        .recency(recencyDimension)
                                        .build();

                        SegmentDemographics segmentDemographics =
SegmentDemographics
                                        .builder()
                                        .build();

                        SegmentLocation segmentLocation = SegmentLocation
                                        .builder()
                                        .build();

                        SegmentDimensions dimensions = SegmentDimensions
                                        .builder()
                                        .attributes(segmentAttributes)
                                        .behavior(segmentBehaviors)
                                        .demographic(segmentDemographics)
                                        .location(segmentLocation)
                                        .build();

                        WriteSegmentRequest writeSegmentRequest =
WriteSegmentRequest.builder()
                                        .name("MySegment")
                                        .dimensions(dimensions)
                                        .build();
```

```
                        CreateSegmentRequest createSegmentRequest =
 CreateSegmentRequest.builder()
                                        .applicationId(appId)
                                        .writeSegmentRequest(writeSegmentRequest)
                                        .build();

                        CreateSegmentResponse createSegmentResult =
 client.createSegment(createSegmentRequest);
                        System.out.println("Segment ID: " +
 createSegmentResult.segmentResponse().id());
                        System.out.println("Done");
                        return createSegmentResult.segmentResponse();

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
                return null;
        }
}
```

- For API details, see CreateSegment in *AWS SDK for Java 2.x API Reference*.

DeleteApp

The following code example shows how to use `DeleteApp`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete an application.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppResponse;
```

```
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteApp {
    public static void main(String[] args) {
        final String usage = """

                Usage:  <appId>

                Where:
                 appId - The ID of the application to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        System.out.println("Deleting an application with ID: " + appId);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deletePinApp(pinpoint, appId);
        System.out.println("Done");
        pinpoint.close();
    }

    public static void deletePinApp(PinpointClient pinpoint, String appId) {
        try {
            DeleteAppRequest appRequest = DeleteAppRequest.builder()
                    .applicationId(appId)
                    .build();

            DeleteAppResponse result = pinpoint.deleteApp(appRequest);
```

```
            String appName = result.applicationResponse().name();
            System.out.println("Application " + appName + " has been deleted.");

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [DeleteApp](#) in *AWS SDK for Java 2.x API Reference*.

`DeleteEndpoint`

The following code example shows how to use `DeleteEndpoint`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an endpoint.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteEndpoint {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appName> <endpointId >

                Where:
                  appId - The id of the application to delete.
                  endpointId - The id of the endpoint to delete.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpointId = args[1];
        System.out.println("Deleting an endpoint with id: " + endpointId);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deletePinEncpoint(pinpoint, appId, endpointId);
        pinpoint.close();
    }

    public static void deletePinEncpoint(PinpointClient pinpoint, String appId,
 String endpointId) {
        try {
            DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .build();

            DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
            String id = result.endpointResponse().id();
            System.out.println("The deleted endpoint id  " + id);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
```

```
}
```

- For API details, see DeleteEndpoint in *AWS SDK for Java 2.x API Reference*.

GetEndpoint

The following code example shows how to use `GetEndpoint`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class LookUpEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId> <endpoint>

                Where:
```

```
                appId - The ID of the application to delete.
                endpoint - The ID of the endpoint.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpoint = args[1];
        System.out.println("Looking up an endpoint point with ID: " + endpoint);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        lookupPinpointEndpoint(pinpoint, appId, endpoint);
        pinpoint.close();
    }

    public static void lookupPinpointEndpoint(PinpointClient pinpoint, String appId,
 String endpoint) {
        try {
            GetEndpointRequest appRequest = GetEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpoint)
                    .build();

            GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
            EndpointResponse endResponse = result.endpointResponse();

            // Uses the Google Gson library to pretty print the endpoint JSON.
            Gson gson = new GsonBuilder()
                    .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                    .setPrettyPrinting()
                    .create();

            String endpointJson = gson.toJson(endResponse);
            System.out.println(endpointJson);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        System.out.println("Done");
    }
}
```

- For API details, see GetEndpoint in *AWS SDK for Java 2.x API Reference*.

GetSegments

The following code example shows how to use `GetSegments`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

List segments.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSegments {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:   <appId>

                Where:
                  appId - The ID of the application that contains a segment.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSegs(pinpoint, appId);
        pinpoint.close();
    }

    public static void listSegs(PinpointClient pinpoint, String appId) {
        try {
            GetSegmentsRequest request = GetSegmentsRequest.builder()
                    .applicationId(appId)
                    .build();

            GetSegmentsResponse response = pinpoint.getSegments(request);
            List<SegmentResponse> segments = response.segmentsResponse().item();
            for (SegmentResponse segment : segments) {
                System.out
                        .println("Segement " + segment.id() + " " + segment.name() +
 " " + segment.lastModifiedDate());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see GetSegments in *AWS SDK for Java 2.x API Reference*.

GetSmsChannel

The following code example shows how to use GetSmsChannel.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/pinpoint#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelResponse;
import software.amazon.awssdk.services.pinpoint.model.GetSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateChannel {
    public static void main(String[] args) {
        final String usage = """

                Usage: CreateChannel <appId>

                Where:
                  appId - The name of the application whose channel is updated.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        SMSChannelResponse getResponse = getSMSChannel(pinpoint, appId);
        toggleSmsChannel(pinpoint, appId, getResponse);
        pinpoint.close();
    }

    private static SMSChannelResponse getSMSChannel(PinpointClient client, String
 appId) {
        try {
            GetSmsChannelRequest request = GetSmsChannelRequest.builder()
                    .applicationId(appId)
                    .build();

            SMSChannelResponse response =
 client.getSmsChannel(request).smsChannelResponse();
            System.out.println("Channel state is " + response.enabled());
            return response;

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void toggleSmsChannel(PinpointClient client, String appId,
 SMSChannelResponse getResponse) {
        boolean enabled = !getResponse.enabled();
        try {
            SMSChannelRequest request = SMSChannelRequest.builder()
                    .enabled(enabled)
                    .build();

            UpdateSmsChannelRequest updateRequest =
 UpdateSmsChannelRequest.builder()
                    .smsChannelRequest(request)
                    .applicationId(appId)
                    .build();
```

```
            UpdateSmsChannelResponse result =
 client.updateSmsChannel(updateRequest);
            System.out.println("Channel state: " +
 result.smsChannelResponse().enabled());

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [GetSmsChannel](#) in *AWS SDK for Java 2.x API Reference*.

`GetUserEndpoints`

The following code example shows how to use `GetUserEndpoints`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListEndpointIds {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <applicationId> <userId>

                Where:
                   applicationId - The ID of the Amazon Pinpoint application that
 has the endpoint.
                   userId - The user id applicable to the endpoints""";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String applicationId = args[0];
        String userId = args[1];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllEndpoints(pinpoint, applicationId, userId);
        pinpoint.close();
    }

    public static void listAllEndpoints(PinpointClient pinpoint,
            String applicationId,
            String userId) {

        try {
            GetUserEndpointsRequest endpointsRequest =
 GetUserEndpointsRequest.builder()
                    .userId(userId)
                    .applicationId(applicationId)
                    .build();

            GetUserEndpointsResponse response =
 pinpoint.getUserEndpoints(endpointsRequest);
            List<EndpointResponse> endpoints = response.endpointsResponse().item();

            // Display the results.
```

```
            for (EndpointResponse endpoint : endpoints) {
                System.out.println("The channel type is: " +
 endpoint.channelType());
                System.out.println("The address is  " + endpoint.address());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see GetUserEndpoints in *AWS SDK for Java 2.x API Reference*.

SendMessages

The following code example shows how to use SendMessages.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Send an email message.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmailPart;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmail;
import software.amazon.awssdk.services.pinpoint.model.EmailMessage;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
```

```
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessage {

        // The character encoding the you want to use for the subject line and
        // message body of the email.
        public static String charset = "UTF-8";

    // The body of the email for recipients whose email clients support HTML
 content.
    static final String body = """
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
 for Java 2.x

        """;

        public static void main(String[] args) {
                final String usage = """

                                Usage:    <subject> <appId> <senderAddress>
 <toAddress>

            Where:
               subject - The email subject to use.
               senderAddress - The from address. This address has to be verified in
 Amazon Pinpoint in the region you're using to send email\s
```

```
              toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String subject = args[0];
        String senderAddress = args[1];
        String toAddress = args[2];
        System.out.println("Sending a message");
        PinpointEmailClient pinpoint = PinpointEmailClient.builder()
            .region(Region.US_EAST_1)
            .build();

        sendEmail(pinpoint, subject, senderAddress, toAddress);
        System.out.println("Email was sent");
        pinpoint.close();
    }

    public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress) {
        try {
            Content content = Content.builder()
                .data(body)
                .build();

            Body messageBody = Body.builder()
                .text(content)
                .build();

            Message message = Message.builder()
                .body(messageBody)
                .subject(Content.builder().data(subject).build())
                .build();

            Destination destination = Destination.builder()
                .toAddresses(toAddress)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(message)
```

```
                .build();

            SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
                .fromEmailAddress(senderAddress)
                .destination(destination)
                .content(emailContent)
                .build();

            pinpointEmailClient.sendEmail(sendEmailRequest);
            System.out.println("Message Sent");

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Send an email message with CC values.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import  software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessageCC {

    // The body of the email.
```

```
    static final String body = """
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
 for Java 2.x

        """;
    public static void main(String[] args) {
        final String usage = """

            Usage:    <subject> <senderAddress> <toAddress> <ccAddress>

            Where:
               subject - The email subject to use.
               senderAddress - The from address. This address has to be verified in
 Amazon Pinpoint in the region you're using to send email\s
               toAddress - The to address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email\s
               ccAddress - The CC address.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String subject = args[0];
        String senderAddress = args[1];
        String toAddress = args[2];
        String ccAddress = args[3];

        System.out.println("Sending a message");
        PinpointEmailClient pinpoint = PinpointEmailClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ArrayList<String> ccList = new ArrayList<>();
        ccList.add(ccAddress);
        sendEmail(pinpoint, subject, senderAddress, toAddress, ccList);
        pinpoint.close();
    }

    public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
 subject, String senderAddress, String toAddress, ArrayList<String> ccAddresses) {
```

```
        try {
            Content content = Content.builder()
                .data(body)
                .build();

            Body messageBody = Body.builder()
                .text(content)
                .build();

            Message message = Message.builder()
                .body(messageBody)
                .subject(Content.builder().data(subject).build())
                .build();

            Destination destination = Destination.builder()
                .toAddresses(toAddress)
                .ccAddresses(ccAddresses)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(message)
                .build();

            SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
                .fromEmailAddress(senderAddress)
                .destination(destination)
                .content(emailContent)
                .build();

            pinpointEmailClient.sendEmail(sendEmailRequest);
            System.out.println("Message Sent");

        } catch (PinpointException e) {
            // Handle exception
            e.printStackTrace();
        }
    }
}
```

Send an SMS message.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessage {

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
        public static String messageType = "TRANSACTIONAL";

        // The registered keyword associated with the originating short code.
        public static String registeredKeyword = "myKeyword";

        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
        // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
        public static String senderId = "MySenderID";

        public static void main(String[] args) {
                final String usage = """

                                Usage:   <message> <appId> <originationNumber>
 <destinationNumber>\s

                                Where:
```

```
                                    message - The body of the message to send.
                                    appId - The Amazon Pinpoint project/application ID
to use when you send this message.
                                    originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
                                    destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s
                                    """;

                if (args.length != 4) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String message = args[0];
                String appId = args[1];
                String originationNumber = args[2];
                String destinationNumber = args[3];
                System.out.println("Sending a message");
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber);
                pinpoint.close();
        }

        public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
                        String originationNumber,
                        String destinationNumber) {
                try {
                        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
                        AddressConfiguration addConfig =
AddressConfiguration.builder()
                                        .channelType(ChannelType.SMS)
                                        .build();

                        addressMap.put(destinationNumber, addConfig);
                        SMSMessage smsMessage = SMSMessage.builder()
```

```
                                        .body(message)
                                        .messageType(messageType)
                                        .originationNumber(originationNumber)
                                        .senderId(senderId)
                                        .keyword(registeredKeyword)
                                        .build();

                        // Create a DirectMessageConfiguration object.
                        DirectMessageConfiguration direct =
 DirectMessageConfiguration.builder()
                                        .smsMessage(smsMessage)
                                        .build();

                        MessageRequest msgReq = MessageRequest.builder()
                                        .addresses(addressMap)
                                        .messageConfiguration(direct)
                                        .build();

                        // create a SendMessagesRequest object
                        SendMessagesRequest request = SendMessagesRequest.builder()
                                        .applicationId(appId)
                                        .messageRequest(msgReq)
                                        .build();

                        SendMessagesResponse response =
 pinpoint.sendMessages(request);
                        MessageResponse msg1 = response.messageResponse();
                        Map map1 = msg1.result();

                        // Write out the result of sendMessage.
                        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

Send batch SMS messages.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageBatch {

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
        public static String messageType = "TRANSACTIONAL";

        // The registered keyword associated with the originating short code.
        public static String registeredKeyword = "myKeyword";

        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
        // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
        public static String senderId = "MySenderID";

        public static void main(String[] args) {
                final String usage = """

                                Usage:   <message> <appId> <originationNumber>
 <destinationNumber> <destinationNumber1>\s

                                Where:
```

```
                                message - The body of the message to send.
                                appId - The Amazon Pinpoint project/application ID
to use when you send this message.
                                originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
                                destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).
                                destinationNumber1 - The second recipient's phone
number.  For best results, you should specify the phone number in E.164 format (for
example, +1-555-555-5654).\s
                                """;

                if (args.length != 5) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String message = args[0];
                String appId = args[1];
                String originationNumber = args[2];
                String destinationNumber = args[3];
                String destinationNumber1 = args[4];
                System.out.println("Sending a message");
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
                pinpoint.close();
        }

        public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
                        String originationNumber,
                        String destinationNumber, String destinationNumber1) {
                try {
                        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
                        AddressConfiguration addConfig =
AddressConfiguration.builder()
                                        .channelType(ChannelType.SMS)
```

```
                                        .build();

                        // Add an entry to the Map object for each number to whom
you want to send a
                        // message.
                        addressMap.put(destinationNumber, addConfig);
                        addressMap.put(destinationNumber1, addConfig);
                        SMSMessage smsMessage = SMSMessage.builder()
                                        .body(message)
                                        .messageType(messageType)
                                        .originationNumber(originationNumber)
                                        .senderId(senderId)
                                        .keyword(registeredKeyword)
                                        .build();

                        // Create a DirectMessageConfiguration object.
                        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
                                        .smsMessage(smsMessage)
                                        .build();

                        MessageRequest msgReq = MessageRequest.builder()
                                        .addresses(addressMap)
                                        .messageConfiguration(direct)
                                        .build();

                        // Create a SendMessagesRequest object.
                        SendMessagesRequest request = SendMessagesRequest.builder()
                                        .applicationId(appId)
                                        .messageRequest(msgReq)
                                        .build();

                        SendMessagesResponse response =
pinpoint.sendMessages(request);
                        MessageResponse msg1 = response.messageResponse();
                        Map map1 = msg1.result();

                        // Write out the result of sendMessage.
                        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
```

```
        }
}
```

- For API details, see [SendMessages](#) in *AWS SDK for Java 2.x API Reference*.

UpdateEndpoint

The following code example shows how to use `UpdateEndpoint`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.EndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.EndpointDemographic;
import software.amazon.awssdk.services.pinpoint.model.EndpointLocation;
import software.amazon.awssdk.services.pinpoint.model.EndpointUser;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.UUID;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Date;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:  <appId>

                Where:
                  appId - The ID of the application to create an endpoint for.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        EndpointResponse response = createEndpoint(pinpoint, appId);
        System.out.println("Got Endpoint: " + response.id());
        pinpoint.close();
    }

    public static EndpointResponse createEndpoint(PinpointClient client, String
 appId) {
        String endpointId = UUID.randomUUID().toString();
        System.out.println("Endpoint ID: " + endpointId);

        try {
            EndpointRequest endpointRequest = createEndpointRequestData();
            UpdateEndpointRequest updateEndpointRequest =
 UpdateEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .endpointRequest(endpointRequest)
                    .build();
```

```
            UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
            System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

            GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .build();

            GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
            System.out.println(getEndpointResponse.endpointResponse().address());

System.out.println(getEndpointResponse.endpointResponse().channelType());

System.out.println(getEndpointResponse.endpointResponse().applicationId());

System.out.println(getEndpointResponse.endpointResponse().endpointStatus());
            System.out.println(getEndpointResponse.endpointResponse().requestId());
            System.out.println(getEndpointResponse.endpointResponse().user());

            return getEndpointResponse.endpointResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static EndpointRequest createEndpointRequestData() {
        try {
            List<String> favoriteTeams = new ArrayList<>();
            favoriteTeams.add("Lakers");
            favoriteTeams.add("Warriors");
            HashMap<String, List<String>> customAttributes = new HashMap<>();
            customAttributes.put("team", favoriteTeams);

            EndpointDemographic demographic = EndpointDemographic.builder()
                    .appVersion("1.0")
                    .make("apple")
                    .model("iPhone")
```

```
                .modelVersion("7")
                .platform("ios")
                .platformVersion("10.1.1")
                .timezone("America/Los_Angeles")
                .build();

        EndpointLocation location = EndpointLocation.builder()
                .city("Los Angeles")
                .country("US")
                .latitude(34.0)
                .longitude(-118.2)
                .postalCode("90068")
                .region("CA")
                .build();

        Map<String, Double> metrics = new HashMap<>();
        metrics.put("health", 100.00);
        metrics.put("luck", 75.00);

        EndpointUser user = EndpointUser.builder()
                .userId(UUID.randomUUID().toString())
                .build();

        DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); // Quoted
 "Z" to indicate UTC, no timezone
                                                                       // offset
        String nowAsISO = df.format(new Date());

        return EndpointRequest.builder()
                .address(UUID.randomUUID().toString())
                .attributes(customAttributes)
                .channelType("APNS")
                .demographic(demographic)
                .effectiveDate(nowAsISO)
                .location(location)
                .metrics(metrics)
                .optOut("NONE")
                .requestId(UUID.randomUUID().toString())
                .user(user)
                .build();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
```

```
        }
        return null;
    }
}
```

- For API details, see UpdateEndpoint in *AWS SDK for Java 2.x API Reference*.

Amazon Pinpoint SMS and Voice API examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Pinpoint SMS and Voice API.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

`SendVoiceMessage`

The following code example shows how to use `SendVoiceMessage`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendVoiceMessage {

        // The Amazon Polly voice that you want to use to send the message. For a
 list
        // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
        static final String voiceName = "Matthew";

        // The language to use when sending the message. For a list of supported
        // languages, see
        // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
        static final String languageCode = "en-US";

        // The content of the message. This example uses SSML to customize and
 control
        // certain aspects of the message, such as by adding pauses and changing
        // phonation. The message can't contain any line breaks.
        static final String ssmlMessage = "<speak>This is a test message sent from "
                        + "<emphasis>Amazon Pinpoint</emphasis> "
                        + "using the <break strength='weak'/>AWS "
                        + "SDK for Java. "
                        + "<amazon:effect phonation='soft'>Thank "
```

```
                        + "you for listening.</amazon:effect></speak>";

        public static void main(String[] args) {

                final String usage = """

                                Usage:   <originationNumber> <destinationNumber>\s

                                Where:
                                  originationNumber - The phone number or short code
 that you specify has to be associated with your Amazon Pinpoint account. For best
 results, specify long codes in E.164 format (for example, +1-555-555-5654).
                                  destinationNumber - The recipient's phone number.
 For best results, you should specify the phone number in E.164 format (for example,
 +1-555-555-5654).\s
                                  """;

                if (args.length != 2) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String originationNumber = args[0];
                String destinationNumber = args[1];
                System.out.println("Sending a voice message");

                // Set the content type to application/json.
                List<String> listVal = new ArrayList<>();
                listVal.add("application/json");
                Map<String, List<String>> values = new HashMap<>();
                values.put("Content-Type", listVal);

                ClientOverrideConfiguration config2 =
 ClientOverrideConfiguration.builder()
                                .headers(values)
                                .build();

                PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
                                .overrideConfiguration(config2)
                                .region(Region.US_EAST_1)
                                .build();

                sendVoiceMsg(client, originationNumber, destinationNumber);
                client.close();
```

```
        }

        public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
 originationNumber,
                        String destinationNumber) {
                try {
                        SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
                                        .languageCode(languageCode)
                                        .text(ssmlMessage)
                                        .voiceId(voiceName)
                                        .build();

                        VoiceMessageContent content = VoiceMessageContent.builder()
                                        .ssmlMessage(ssmlMessageType)
                                        .build();

                        SendVoiceMessageRequest voiceMessageRequest =
 SendVoiceMessageRequest.builder()
                                        .destinationPhoneNumber(destinationNumber)
                                        .originationPhoneNumber(originationNumber)
                                        .content(content)
                                        .build();

                        client.sendVoiceMessage(voiceMessageRequest);
                        System.out.println("The message was sent successfully.");

                } catch (PinpointSmsVoiceException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

- For API details, see SendVoiceMessage in *AWS SDK for Java 2.x API Reference*.

Amazon Polly examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Polly.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

`DescribeVoices`

The following code example shows how to use `DescribeVoices`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.Voice;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeVoicesSample {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        describeVoice(polly);
        polly.close();
    }

    public static void describeVoice(PollyClient polly) {
        try {
            DescribeVoicesRequest voicesRequest = DescribeVoicesRequest.builder()
                    .languageCode("en-US")
                    .build();

            DescribeVoicesResponse enUsVoicesResult =
 polly.describeVoices(voicesRequest);
            List<Voice> voices = enUsVoicesResult.voices();
            for (Voice myVoice : voices) {
                System.out.println("The ID of the voice is " + myVoice.id());
                System.out.println("The gender of the voice is " +
 myVoice.gender());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- For API details, see [DescribeVoices](#) in *AWS SDK for Java 2.x API Reference*.

ListLexicons

The following code example shows how to use `ListLexicons`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.ListLexiconsResponse;
import software.amazon.awssdk.services.polly.model.ListLexiconsRequest;
import software.amazon.awssdk.services.polly.model.LexiconDescription;
import software.amazon.awssdk.services.polly.model.PollyException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLexicons {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        listLexicons(polly);
        polly.close();
    }

    public static void listLexicons(PollyClient client) {
        try {
            ListLexiconsRequest listLexiconsRequest = ListLexiconsRequest.builder()
                    .build();

            ListLexiconsResponse listLexiconsResult =
 client.listLexicons(listLexiconsRequest);
```

```
            List<LexiconDescription> lexiconDescription =
 listLexiconsResult.lexicons();
            for (LexiconDescription lexDescription : lexiconDescription) {
                System.out.println("The name of the Lexicon is " +
 lexDescription.name());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- For API details, see ListLexicons in *AWS SDK for Java 2.x API Reference*.

SynthesizeSpeech

The following code example shows how to use `SynthesizeSpeech`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import javazoom.jl.decoder.JavaLayerException;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.Voice;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.OutputFormat;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechRequest;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechResponse;
import java.io.IOException;
import java.io.InputStream;
```

```
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PollyDemo {
    private static final String SAMPLE = "Congratulations. You have successfully
 built this working demo " +
            " of Amazon Polly in Java Version 2. Have fun building voice enabled
 apps with Amazon Polly (that's me!), and always "
            +
            " look at the AWS website for tips and tricks on using Amazon Polly and
 other great services from AWS";

    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        talkPolly(polly);
        polly.close();
    }

    public static void talkPolly(PollyClient polly) {
        try {
            DescribeVoicesRequest describeVoiceRequest =
 DescribeVoicesRequest.builder()
                    .engine("standard")
                    .build();

            DescribeVoicesResponse describeVoicesResult =
 polly.describeVoices(describeVoiceRequest);
            Voice voice = describeVoicesResult.voices().stream()
                    .filter(v -> v.name().equals("Joanna"))
                    .findFirst()
                    .orElseThrow(() -> new RuntimeException("Voice not found"));
            InputStream stream = synthesize(polly, SAMPLE, voice, OutputFormat.MP3);
```

```
            AdvancedPlayer player = new AdvancedPlayer(stream,

 javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());
            player.setPlayBackListener(new PlaybackListener() {
                public void playbackStarted(PlaybackEvent evt) {
                    System.out.println("Playback started");
                    System.out.println(SAMPLE);
                }

                public void playbackFinished(PlaybackEvent evt) {
                    System.out.println("Playback finished");
                }
            });

            // play it!
            player.play();

        } catch (PollyException | JavaLayerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static InputStream synthesize(PollyClient polly, String text, Voice
 voice, OutputFormat format)
            throws IOException {
        SynthesizeSpeechRequest synthReq = SynthesizeSpeechRequest.builder()
                .text(text)
                .voiceId(voice.id())
                .outputFormat(format)
                .build();

        ResponseInputStream<SynthesizeSpeechResponse> synthRes =
 polly.synthesizeSpeech(synthReq);
        return synthRes;
    }
}
```

- For API details, see SynthesizeSpeech in *AWS SDK for Java 2.x API Reference*.

Amazon RDS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon RDS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon RDS

The following code examples show how to get started using Amazon RDS.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " + instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DescribeDBInstances in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios
- Serverless examples

Actions

CreateDBInstance

The following code example shows how to use `CreateDBInstance`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/rds#code-examples).

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
 *
```

```
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 *
 */

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbName> <secretName>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    dbName - The database name.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials."
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
 user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }
```

```
    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbName,
            String userName,
            String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .allocatedStorage(100)
                    .dbName(dbName)
                    .engine("mysql")
                    .dbInstanceClass("db.m4.large")
                    .engineVersion("8.0")
                    .storageType("standard")
                    .masterUsername(userName)
                    .masterUserPassword(userPassword)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            // Loop until the cluster is ready.
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available"))
                        instanceReady = true;
                    else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

CreateDBParameterGroup

The following code example shows how to use `CreateDBParameterGroup`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/rds#code-examples).

```
    public static void createDBParameterGroup(RdsClient rdsClient, String
 dbGroupName, String dbParameterGroupFamily) {
        try {
            CreateDbParameterGroupRequest groupRequest =
 CreateDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbParameterGroupResponse response =
 rdsClient.createDBParameterGroup(groupRequest);
            System.out.println("The group name is " +
 response.dbParameterGroup().dbParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see [CreateDBParameterGroup](https://docs.aws.amazon.com/goto/SdkForJavaV2/rds-2014-10-31/CreateDBParameterGroup) in *AWS SDK for Java 2.x API Reference*.

CreateDBSnapshot

The following code example shows how to use `CreateDBSnapshot`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    // Create an Amazon RDS snapshot.
    public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
        try {
            CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
            System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see [CreateDBSnapshot](#) in *AWS SDK for Java 2.x API Reference*.

`DeleteDBInstance`

The following code example shows how to use `DeleteDBInstance`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
```

```
                .build();

        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
 DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
 response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteDBInstance in *AWS SDK for Java 2.x API Reference*.

`DeleteDBParameterGroup`

The following code example shows how to use `DeleteDBParameterGroup`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(dbARN) == 0) {
                        System.out.println(dbARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DeleteDBParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeAccountAttributes`

The following code example shows how to use `DescribeAccountAttributes`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
```

```
                .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
            DescribeAccountAttributesResponse response =
 rdsClient.describeAccountAttributes();
            List<AccountQuota> quotasList = response.accountQuotas();
            for (AccountQuota quotas : quotasList) {
                System.out.println("Name is: " + quotas.accountQuotaName());
                System.out.println("Max value is " + quotas.max());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DescribeAccountAttributes in *AWS SDK for Java 2.x API Reference*.

DescribeDBEngineVersions

The following code example shows how to use `DescribeDBEngineVersions`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
 DescribeDbEngineVersionsRequest.builder()
```

```
                    .defaultOnly(true)
                    .engine("mysql")
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
engineOb.engine());
                System.out.println("The version number of the database engine " +
engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for Java 2.x API Reference*.

DescribeDBInstances

The following code example shows how to use `DescribeDBInstances`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " + instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DescribeDBInstances in *AWS SDK for Java 2.x API Reference*.

DescribeDBParameterGroups

The following code example shows how to use `DescribeDBParameterGroups`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeDbParameterGroups(RdsClient rdsClient, String
 dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
 DescribeDbParameterGroupsRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .maxRecords(20)
                    .build();

            DescribeDbParameterGroupsResponse response =
 rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
                System.out.println("The group name is " +
 group.dbParameterGroupName());
                System.out.println("The group description is " +
 group.description());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DescribeDBParameterGroups](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeDBParameters`

The following code example shows how to use `DescribeDBParameters`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    // Retrieve parameters in the group.
    public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
 int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbParametersResponse response =
 rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") == 0)) {
                    System.out.println("*** The parameter name is  " + paraName);
```

```
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DescribeDBParameters](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeOrderableDBInstanceOptions`

The following code example shows how to use `DescribeOrderableDBInstanceOptions`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("mysql")
                    .build();
```

```
            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeOrderableDBInstanceOptions in *AWS SDK for Java 2.x API Reference*.

`GenerateRDSAuthToken`

The following code example shows how to use `GenerateRDSAuthToken`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Use the RdsUtilities class to generate an authentication token.

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <masterUsername>

                Where:
```

```
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUsername - The master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUsername = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        String token = getAuthToken(rdsClient, dbInstanceIdentifier,
 masterUsername);
        System.out.println("The token response is " + token);
    }

    public static String getAuthToken(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUsername) {

        RdsUtilities utilities = rdsClient.utilities();
        try {
            GenerateAuthenticationTokenRequest tokenRequest =
 GenerateAuthenticationTokenRequest.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .username(masterUsername)
                    .port(3306)
                    .hostname(dbInstanceIdentifier)
                    .build();

            return utilities.generateAuthenticationToken(tokenRequest);

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see GenerateRDSAuthToken in *AWS SDK for Java 2.x API Reference*.

ModifyDBInstance

The following code example shows how to use `ModifyDBInstance`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUserPassword - The updated password that corresponds to
 the master user name.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUserPassword = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
        rdsClient.close();
    }

    public static void updateIntance(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUserPassword) {
        try {
            // For a demo - modify the DB instance by modifying the master password.
            ModifyDbInstanceRequest modifyDbInstanceRequest =
 ModifyDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .publiclyAccessible(true)
                    .masterUserPassword(masterUserPassword)
                    .build();

            ModifyDbInstanceResponse instanceResponse =
 rdsClient.modifyDBInstance(modifyDbInstanceRequest);
            System.out.print("The ARN of the modified database is: " +
 instanceResponse.dbInstance().dbInstanceArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ModifyDBInstance in *AWS SDK for Java 2.x API Reference*.

ModifyDBParameterGroup

The following code example shows how to use `ModifyDBParameterGroup`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbParameterGroupRequest groupRequest =
 ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbParameterGroupResponse response =
 rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
 response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see ModifyDBParameterGroup in *AWS SDK for Java 2.x API Reference*.

RebootDBInstance

The following code example shows how to use `RebootDBInstance`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/rds#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
```

```
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void rebootInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            RebootDbInstanceRequest rebootDbInstanceRequest =
 RebootDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            RebootDbInstanceResponse instanceResponse =
 rdsClient.rebootDBInstance(rebootDbInstanceRequest);
            System.out.print("The database " +
 instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see RebootDBInstance in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with DB instances

The following code example shows how to:

- Create a custom DB parameter group and set parameter values.
- Create a DB instance that's configured to use the parameter group. The DB instance also contains a database.
- Take a snapshot of the instance.

- Delete the instance and parameter group.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run multiple operations.

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
```

```
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs these tasks:
 *
 * 1. Returns a list of the available DB engines.
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySql database and uses
 * the parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection
 * endpoint value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
```

```
 * 13. Deletes the RDS DB instance.
 * 14. Deletes the parameter group.
 */
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
 <dbName> <dbSnapshotIdentifier> <secretName>

                Where:
                    dbGroupName - The database group name.\s
                    dbParameterGroupFamily - The database parameter group name (for
 example, mysql8.0).
                    dbInstanceIdentifier - The database instance identifier\s
                    dbName - The database name.\s
                    dbSnapshotIdentifier - The snapshot identifier.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials"
                """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceIdentifier = args[2];
        String dbName = args[3];
        String dbSnapshotIdentifier = args[4];
        String secretName = args[5];

        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        String masterUsername = user.getUsername();
        String masterUserPassword = user.getPassword();

        Region region = Region.US_WEST_2;
```

```
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon RDS example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbParameterGroups(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbParameters(rdsClient, dbGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBParas(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated value");
        describeDbParameters(rdsClient, dbGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("8. Get a list of micro instance classes available for
 the selected engine");
        getMicroInstances(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "9. Create an RDS database instance that contains a MySql database
 and uses the parameter group");
        String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
 dbInstanceIdentifier, dbName, masterUsername,
                masterUserPassword);
        System.out.println("The ARN of the new database is " + dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a snapshot of the DB instance");
        createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the parameter group");
        deleteParaGroup(rdsClient, dbGroupName, dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
```

```
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    public static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
```

```
                    if (instanceARN.compareTo(dbARN) == 0) {
                        System.out.println(dbARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
 DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Delete the DB instance.
    public static void deleteDatabaseInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
 DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
 response.dbInstance().dbInstanceStatus());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the snapshot instance is available.
    public static void waitForSnapshotReady(RdsClient rdsClient, String
 dbInstanceIdentifier,
            String dbSnapshotIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbSnapshotsRequest snapshotsRequest =
 DescribeDbSnapshotsRequest.builder()
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbSnapshotsResponse response =
 rdsClient.describeDBSnapshots(snapshotsRequest);
                List<DBSnapshot> snapshotList = response.dbSnapshots();
                for (DBSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");
        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
    // Create an Amazon RDS snapshot.
    public static void createSnapshot(RdsClient rdsClient, String
 dbInstanceIdentifier, String dbSnapshotIdentifier) {
        try {
            CreateDbSnapshotRequest snapshotRequest =
 CreateDbSnapshotRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbSnapshotResponse response =
 rdsClient.createDBSnapshot(snapshotRequest);
            System.out.println("The Snapshot id is " +
 response.dbSnapshot().dbiResourceId());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
 DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
```

```
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Create a database instance and return the ARN of the database.
    public static String createDatabaseInstance(RdsClient rdsClient,
            String dbGroupName,
            String dbInstanceIdentifier,
            String dbName,
            String masterUsername,
            String masterUserPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .allocatedStorage(100)
                    .dbName(dbName)
                    .dbParameterGroupName(dbGroupName)
                    .engine("mysql")
                    .dbInstanceClass("db.m4.large")
                    .engineVersion("8.0")
                    .storageType("standard")
                    .masterUsername(masterUsername)
                    .masterUserPassword(masterUserPassword)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }

        return "";
    }

    // Get a list of micro instances.
    public static void getMicroInstances(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
 DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("mysql")
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
            List<OrderableDBInstanceOption> orderableDBInstances =
 response.orderableDBInstanceOptions();
            for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
 {
                System.out.println("The engine version is " +
 dbInstanceOption.engineVersion());
                System.out.println("The engine description is " +
 dbInstanceOption.engine());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
 dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
 DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("mysql")
                    .build();
```

```
            DescribeDbEngineVersionsResponse response =
 rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
 dbEngine.engineVersion());
                System.out.println("The engine description is " +
 dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbParameterGroupRequest groupRequest =
 ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbParameterGroupResponse response =
 rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
 response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
    // Retrieve parameters in the group.
    public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
 int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbParametersResponse response =
 rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") == 0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
 para.parameterValue());
                    System.out.println("*** The parameter data type is " +
 para.dataType());
                    System.out.println("*** The parameter description is " +
 para.description());
                    System.out.println("*** The parameter allowed values  is " +
 para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
    public static void describeDbParameterGroups(RdsClient rdsClient, String
 dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
 DescribeDbParameterGroupsRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .maxRecords(20)
                    .build();

            DescribeDbParameterGroupsResponse response =
 rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
                System.out.println("The group name is " +
 group.dbParameterGroupName());
                System.out.println("The group description is " +
 group.description());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBParameterGroup(RdsClient rdsClient, String
 dbGroupName, String dbParameterGroupFamily) {
        try {
            CreateDbParameterGroupRequest groupRequest =
 CreateDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbParameterGroupResponse response =
 rdsClient.createDBParameterGroup(groupRequest);
            System.out.println("The group name is " +
 response.dbParameterGroup().dbParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
```

```
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
 DescribeDbEngineVersionsRequest.builder()
                    .defaultOnly(true)
                    .engine("mysql")
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
 rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions

- DescribeDBInstances
- DescribeDBParameterGroups
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Serverless examples

Connecting to an Amazon RDS database in a Lambda function

The following code example shows how to implement a Lambda function that connects to an RDS database. The function makes a simple database request and returns the result.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Connecting to an Amazon RDS database in a Lambda function using Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

```
public class RdsLambdaHandler implements RequestHandler<APIGatewayProxyRequestEvent,
 APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new APIGatewayProxyResponseEvent();

        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"), token);
                 PreparedStatement statement = connection.prepareStatement("SELECT ?
 + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
```

```
    }

    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();

        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response = rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

Amazon Redshift examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Redshift.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon Redshift

The following code examples show how to get started using Amazon Redshift.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/redshift#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.paginators.DescribeClustersIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloRedshift {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        listClustersPaginator(redshiftClient);
    }

    public static void listClustersPaginator(RedshiftClient redshiftClient) {
        DescribeClustersIterable clustersIterable =
 redshiftClient.describeClustersPaginator();
        clustersIterable.stream()
            .flatMap(r -> r.clusters().stream())
            .forEach(cluster -> System.out
                .println(" Cluster identifier: " + cluster.clusterIdentifier() + "
 status = " + cluster.clusterStatus()));
```

```
    }
}
```

- For API details, see [describeClusters](#) in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

`CreateCluster`

The following code example shows how to use `CreateCluster`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the cluster.

```
    public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
                                     String masterUserPassword) {
        try {
            CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
                .clusterIdentifier(clusterId)
                .masterUsername(masterUsername)
                .masterUserPassword(masterUserPassword)
                .nodeType("ra3.4xlarge")
                .publiclyAccessible(true)
                .numberOfNodes(2)
                .build();
```

```
            CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
            System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());

        } catch (RedshiftException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see CreateCluster in *AWS SDK for Java 2.x API Reference*.

`CreateTable`

The following code example shows how to use `CreateTable`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void createTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName) {
        try {
            ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
                .clusterIdentifier(clusterId)
                .dbUser(userName)
                .database(databaseName)
                .sql("CREATE TABLE Movies ("
                    + "id INT PRIMARY KEY, "
                    + "title VARCHAR(100), "
                    + "year INT)")
                .build();

            redshiftDataClient.executeStatement(createTableRequest);
```

```
            System.out.println("Table created: Movies");

        } catch (RedshiftDataException e) {
            System.err.println("Error creating table: " + e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see [CreateTable](#) in *AWS SDK for Java 2.x API Reference*.

DeleteCluster

The following code example shows how to use `DeleteCluster`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete the cluster.

```
    public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
 clusterId) {
        try {
            DeleteClusterRequest deleteClusterRequest =
 DeleteClusterRequest.builder()
                .clusterIdentifier(clusterId)
                .skipFinalClusterSnapshot(true)
                .build();

            DeleteClusterResponse response =
 redshiftClient.deleteCluster(deleteClusterRequest);
            System.out.println("The status is " +
 response.cluster().clusterStatus());

        } catch (RedshiftException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
```

- For API details, see [DeleteCluster](#) in *AWS SDK for Java 2.x API Reference*.

DescribeClusters

The following code example shows how to use `DescribeClusters`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Describe the cluster.

```
    public static void waitForClusterReady(RedshiftClient redshiftClient, String 
 clusterId) {
        boolean clusterReady = false;
        String clusterReadyStr;
        System.out.println("Waiting for cluster to become available. This may take a 
 few mins.");
        try {
            DescribeClustersRequest clustersRequest = 
 DescribeClustersRequest.builder()
                .clusterIdentifier(clusterId)
                .build();
            long startTime = System.currentTimeMillis();

            // Loop until the cluster is ready.
            while (!clusterReady) {
                DescribeClustersResponse clusterResponse = 
 redshiftClient.describeClusters(clustersRequest);
                List<Cluster> clusterList = clusterResponse.clusters();
                for (Cluster cluster : clusterList) {
                    clusterReadyStr = cluster.clusterStatus();
                    if (clusterReadyStr.contains("available"))
                        clusterReady = true;
```

```
                    else {
                        long elapsedTimeMillis = System.currentTimeMillis() -
startTime;
                        long elapsedSeconds = elapsedTimeMillis / 1000;
                        long minutes = elapsedSeconds / 60;
                        long seconds = elapsedSeconds % 60;

                        System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
                        TimeUnit.SECONDS.sleep(5);
                    }
                }
            }

            long elapsedTimeMillis = System.currentTimeMillis() - startTime;
            long elapsedSeconds = elapsedTimeMillis / 1000;
            long minutes = elapsedSeconds / 60;
            long seconds = elapsedSeconds % 60;

            System.out.println(String.format("Cluster is available! Total Elapsed
Time: %02d:%02d", minutes, seconds));

        } catch (RedshiftException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeClusters in *AWS SDK for Java 2.x API Reference*.

DescribeStatement

The following code example shows how to use `DescribeStatement`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void checkStatement(RedshiftDataClient redshiftDataClient, String
 sqlId) {
        try {
            DescribeStatementRequest statementRequest =
 DescribeStatementRequest.builder()
                .id(sqlId)
                .build();

            String status;
            while (true) {
                DescribeStatementResponse response =
 redshiftDataClient.describeStatement(statementRequest);
                status = response.statusAsString();
                System.out.println("..." + status);

                if (status.compareTo("FAILED") == 0 ) {
                    System.out.println("The Query Failed. Ending program");
                    System.exit(1);

                } else if (status.compareTo("FINISHED") == 0) {
                    break;
                }
                TimeUnit.SECONDS.sleep(1);
            }

            System.out.println("The statement is finished!");

        } catch (RedshiftDataException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DescribeStatement](#) in *AWS SDK for Java 2.x API Reference*.

GetStatementResult

The following code example shows how to use `GetStatementResult`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Check the statement result.

```
   public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
        try {
            GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
                .id(statementId)
                .build();

            // Extract and print the field values using streams.
            GetStatementResultResponse response =
redshiftDataClient.getStatementResult(resultRequest);
            response.records().stream()
                .flatMap(List::stream)
                .map(Field::stringValue)
                .filter(value -> value != null)
                .forEach(value -> System.out.println("The Movie title field is " +
value));

        } catch (RedshiftDataException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see [GetStatementResult](#) in *AWS SDK for Java 2.x API Reference*.

`Insert`

The following code example shows how to use `Insert`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/redshift#code-examples).

```
    public static void popTable(RedshiftDataClient redshiftDataClient, String
 clusterId, String databaseName, String userName, String fileName, int number)
 throws IOException {
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            if (t == number)
                break;
            currentNode = (ObjectNode) iter.next();
            int year = currentNode.get("year").asInt();
            String title = currentNode.get("title").asText();

            // Use SqlParameter to avoid SQL injection.
            List<SqlParameter> parameterList = new ArrayList<>();
            String sqlStatement = "INSERT INTO Movies
 VALUES( :id , :title, :year);";

            // Create the parameters.
            SqlParameter idParam = SqlParameter.builder()
                .name("id")
                .value(String.valueOf(t))
                .build();

            SqlParameter titleParam= SqlParameter.builder()
                .name("title")
                .value(title)
                .build();

            SqlParameter yearParam = SqlParameter.builder()
                .name("year")
```

```
                .value(String.valueOf(year))
                .build();
            parameterList.add(idParam);
            parameterList.add(titleParam);
            parameterList.add(yearParam);

            try {
                ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                    .clusterIdentifier(clusterId)
                    .sql(sqlStatement)
                    .database(databaseName)
                    .dbUser(userName)
                    .parameters(parameterList)
                    .build();

                redshiftDataClient.executeStatement(insertStatementRequest);
                System.out.println("Inserted: " + title + " (" + year + ")");
                t++;

            } catch (RedshiftDataException e) {
                System.err.println("Error inserting data: " + e.getMessage());
                System.exit(1);
            }
        }
        System.out.println(t + " records were added to the Movies table. ");
    }
```

- For API details, see Insert in *AWS SDK for Java 2.x API Reference*.

`ModifyCluster`

The following code example shows how to use `ModifyCluster`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Modify a cluster.

```
    public static void modifyCluster(RedshiftClient redshiftClient, String
 clusterId) {
        try {
            ModifyClusterRequest modifyClusterRequest =
 ModifyClusterRequest.builder()
                .clusterIdentifier(clusterId)
                .preferredMaintenanceWindow("wed:07:30-wed:08:00")
                .build();

            ModifyClusterResponse clusterResponse =
 redshiftClient.modifyCluster(modifyClusterRequest);
            System.out.println("The modified cluster was successfully modified and
 has "
                + clusterResponse.cluster().preferredMaintenanceWindow() + " as the
 maintenance window");

        } catch (RedshiftException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ModifyCluster in *AWS SDK for Java 2.x API Reference*.

Query

The following code example shows how to use `Query`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Query a table.

```
    public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
```

```
                                                    String database,
                                                    String dbUser,
                                                    int year,
                                                    String clusterId) {

        try {
            String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
            SqlParameter yearParam= SqlParameter.builder()
                .name("year")
                .value(String.valueOf(year))
                .build();

            ExecuteStatementRequest statementRequest =
 ExecuteStatementRequest.builder()
                .clusterIdentifier(clusterId)
                .database(database)
                .dbUser(dbUser)
                .parameters(yearParam)
                .sql(sqlStatement)
                .build();

            ExecuteStatementResponse response =
 redshiftDataClient.executeStatement(statementRequest);
            return response.id();

        } catch (RedshiftDataException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see Query in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with Amazon Redshift

The following code example shows how to work with Amazon Redshift tables, items, and queries.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ObjectNode;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.model.Cluster;
import software.amazon.awssdk.services.redshift.model.CreateClusterRequest;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterRequest;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.DescribeClustersRequest;
import software.amazon.awssdk.services.redshift.model.DescribeClustersResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterRequest;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import software.amazon.awssdk.services.redshiftdata.RedshiftDataClient;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.Field;
import software.amazon.awssdk.services.redshiftdata.model.GetStatementResultRequest;
import
 software.amazon.awssdk.services.redshiftdata.model.GetStatementResultResponse;
import software.amazon.awssdk.services.redshiftdata.model.ListDatabasesRequest;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import software.amazon.awssdk.services.redshiftdata.model.SqlParameter;
import
 software.amazon.awssdk.services.redshiftdata.paginators.ListDatabasesIterable;
import com.fasterxml.jackson.core.JsonParser;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
```

```
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *
 This Java example performs these tasks:
 *
 * 1. Prompts the user for a unique cluster ID or use the default value.
 * 2. Creates a Redshift cluster with the specified or default cluster Id value.
 * 3. Waits until the Redshift cluster is available for use.
 * 4. Lists all databases using a pagination API call.
 * 5. Creates a table named "Movies" with fields ID, title, and year.
 * 6. Inserts a specified number of records into the "Movies" table by reading the
 Movies JSON file.
 * 7. Prompts the user for a movie release year.
 * 8. Runs a SQL query to retrieve movies released in the specified year.
 * 9. Modifies the Redshift cluster.
 * 10. Prompts the user for confirmation to delete the Redshift cluster.
 * 11. If confirmed, deletes the specified Redshift cluster.
 */

public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <jsonFilePath>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate that
 file in ../../../resources/sample_files/movies.json)
            """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String jsonFilePath = args[0];
        String userName;
        String userPassword;
        String databaseName = "dev" ;
        Scanner scanner = new Scanner(System.in);

        Region region = Region.US_EAST_1;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        RedshiftDataClient redshiftDataClient = RedshiftDataClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Redshift SDK Getting Started 
scenario.");
        System.out.println("""
        This Java program demonstrates how to interact with Amazon Redshift by using 
the AWS SDK for Java (v2).\s
        Amazon Redshift is a fully managed, petabyte-scale data warehouse service 
hosted in the cloud.

        The program's primary functionalities include cluster creation, verification 
of cluster readiness,\s
        list databases, table creation, data population within the table, and 
execution of SQL statements.
        Furthermore, it demonstrates the process of querying data from the Movie 
table.\s

        Upon completion of the program, all AWS resources are cleaned up.
        """);

        System.out.println("Lets get started...");
        System.out.println("Please enter your user name (default is awsuser)");
        String user = scanner.nextLine();
        userName = user.isEmpty() ? "awsuser" : user;
        System.out.println(DASHES);
```

```
        System.out.println("Please enter your user password (default is
 AwsUser1000)");
        String userpass = scanner.nextLine();
        userPassword = userpass.isEmpty() ? "AwsUser1000" : userpass;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A Redshift cluster refers to the collection of computing
 resources and storage that work together to process and analyze large volumes of
 data.");
        System.out.println("Enter a cluster id value (default is redshift-cluster-
movies): ");
        String userClusterId = scanner.nextLine();
        String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
 userClusterId;
        createCluster(redshiftClient, clusterId, userName, userPassword);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Wait until "+clusterId +" is available.");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        waitForClusterReady(redshiftClient, clusterId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        String databaseInfo = """
            When you created $clusteridD, the dev database is created by default and
 used in this scenario.\s

            To create a custom database, you need to have a CREATEDB privilege.\s
            For more information, see the documentation here: https://
docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
           """.replace("$clusteridD", clusterId);

        System.out.println(databaseInfo);
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("List databases in "+clusterId);
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
```

```
        listAllDatabases(redshiftDataClient, clusterId, userName, databaseName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Now you will create a table named Movies.");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        createTable(redshiftDataClient, clusterId, databaseName, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Populate the Movies table using the Movies.json file.");
        System.out.println("Specify the number of records you would like to add to 
 the Movies Table.");
        System.out.println("Please enter a value between 50 and 200.");
        int numRecords;
        do {
            System.out.print("Enter a value: ");
            while (!scanner.hasNextInt()) {
                System.out.println("Invalid input. Please enter a value between 50 
 and 200.");
                System.out.print("Enter a year: ");
                scanner.next();
            }
            numRecords = scanner.nextInt();
        } while (numRecords < 50 || numRecords > 200);
        popTable(redshiftDataClient, clusterId, databaseName, userName, 
 jsonFilePath, numRecords);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Query the Movies table by year. Enter a value between 
 2012-2014.");
        int movieYear;
        do {
            System.out.print("Enter a year: ");
            while (!scanner.hasNextInt()) {
                System.out.println("Invalid input. Please enter a valid year between 
 2012 and 2014.");
                System.out.print("Enter a year: ");
                scanner.next();
            }
            movieYear = scanner.nextInt();
            scanner.nextLine();
```

```
        } while (movieYear < 2012 || movieYear > 2014);

        String id = queryMoviesByYear(redshiftDataClient, databaseName, userName,
 movieYear, clusterId);
        System.out.println("The identifier of the statement is " + id);
        checkStatement(redshiftDataClient, id);
        getResults(redshiftDataClient, id);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Now you will modify the Redshift cluster.");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        modifyCluster(redshiftClient, clusterId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to delete the Amazon Redshift cluster?
 (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete " +clusterId);
            System.out.print("Press Enter to continue...");
            scanner.nextLine();
            deleteRedshiftCluster(redshiftClient, clusterId);
        } else {
            System.out.println("The "+clusterId +" was not deleted");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This concludes the Amazon Redshift SDK Getting Started
 scenario.");
        System.out.println(DASHES);
    }

    public static void listAllDatabases(RedshiftDataClient redshiftDataClient,
 String clusterId, String dbUser, String database) {
        try {
            ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
                .clusterIdentifier(clusterId)
                .dbUser(dbUser)
                .database(database)
                .build();
```

```
            ListDatabasesIterable listDatabasesIterable =
redshiftDataClient.listDatabasesPaginator(databasesRequest);
            listDatabasesIterable.stream()
                .flatMap(r -> r.databases().stream())
                .forEach(db -> System.out
                    .println("The database name is : " + db));

        } catch (RedshiftDataException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {
        try {
            DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
                .clusterIdentifier(clusterId)
                .skipFinalClusterSnapshot(true)
                .build();

            DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
            System.out.println("The status is " +
response.cluster().clusterStatus());

        } catch (RedshiftException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void popTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName, String fileName, int number)
throws IOException {
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
```

```
            if (t == number)
                break;
            currentNode = (ObjectNode) iter.next();
            int year = currentNode.get("year").asInt();
            String title = currentNode.get("title").asText();

            // Use SqlParameter to avoid SQL injection.
            List<SqlParameter> parameterList = new ArrayList<>();
            String sqlStatement = "INSERT INTO Movies
 VALUES( :id , :title, :year);";

            // Create the parameters.
            SqlParameter idParam = SqlParameter.builder()
                .name("id")
                .value(String.valueOf(t))
                .build();

            SqlParameter titleParam= SqlParameter.builder()
                .name("title")
                .value(title)
                .build();

            SqlParameter yearParam = SqlParameter.builder()
                .name("year")
                .value(String.valueOf(year))
                .build();
            parameterList.add(idParam);
            parameterList.add(titleParam);
            parameterList.add(yearParam);

            try {
                ExecuteStatementRequest insertStatementRequest =
 ExecuteStatementRequest.builder()
                    .clusterIdentifier(clusterId)
                    .sql(sqlStatement)
                    .database(databaseName)
                    .dbUser(userName)
                    .parameters(parameterList)
                    .build();

                redshiftDataClient.executeStatement(insertStatementRequest);
                System.out.println("Inserted: " + title + " (" + year + ")");
                t++;
```

```
            } catch (RedshiftDataException e) {
                System.err.println("Error inserting data: " + e.getMessage());
                System.exit(1);
            }
        }
        System.out.println(t + " records were added to the Movies table. ");
    }

    public static void checkStatement(RedshiftDataClient redshiftDataClient, String
 sqlId) {
        try {
            DescribeStatementRequest statementRequest =
 DescribeStatementRequest.builder()
                .id(sqlId)
                .build();

            String status;
            while (true) {
                DescribeStatementResponse response =
 redshiftDataClient.describeStatement(statementRequest);
                status = response.statusAsString();
                System.out.println("..." + status);

                if (status.compareTo("FAILED") == 0 ) {
                    System.out.println("The Query Failed. Ending program");
                    System.exit(1);

                } else if (status.compareTo("FINISHED") == 0) {
                    break;
                }
                TimeUnit.SECONDS.sleep(1);
            }

            System.out.println("The statement is finished!");

        } catch (RedshiftDataException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void modifyCluster(RedshiftClient redshiftClient, String
 clusterId) {
        try {
```

```
            ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
                .clusterIdentifier(clusterId)
                .preferredMaintenanceWindow("wed:07:30-wed:08:00")
                .build();

            ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
            System.out.println("The modified cluster was successfully modified and
has "
                + clusterResponse.cluster().preferredMaintenanceWindow() + " as the
maintenance window");

        } catch (RedshiftException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
                                           String database,
                                           String dbUser,
                                           int year,
                                           String clusterId) {

        try {
            String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
            SqlParameter yearParam= SqlParameter.builder()
                .name("year")
                .value(String.valueOf(year))
                .build();

            ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
                .clusterIdentifier(clusterId)
                .database(database)
                .dbUser(dbUser)
                .parameters(yearParam)
                .sql(sqlStatement)
                .build();

            ExecuteStatementResponse response =
redshiftDataClient.executeStatement(statementRequest);
            return response.id();
```

```
        } catch (RedshiftDataException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
        try {
            GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
                .id(statementId)
                .build();

            // Extract and print the field values using streams.
            GetStatementResultResponse response =
redshiftDataClient.getStatementResult(resultRequest);
            response.records().stream()
                .flatMap(List::stream)
                .map(Field::stringValue)
                .filter(value -> value != null)
                .forEach(value -> System.out.println("The Movie title field is " +
value));

        } catch (RedshiftDataException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void waitForClusterReady(RedshiftClient redshiftClient, String
clusterId) {
        boolean clusterReady = false;
        String clusterReadyStr;
        System.out.println("Waiting for cluster to become available. This may take a
few mins.");
        try {
            DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
                .clusterIdentifier(clusterId)
                .build();
            long startTime = System.currentTimeMillis();
```

```
            // Loop until the cluster is ready.
            while (!clusterReady) {
                DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters(clustersRequest);
                List<Cluster> clusterList = clusterResponse.clusters();
                for (Cluster cluster : clusterList) {
                    clusterReadyStr = cluster.clusterStatus();
                    if (clusterReadyStr.contains("available"))
                        clusterReady = true;
                    else {
                        long elapsedTimeMillis = System.currentTimeMillis() -
startTime;
                        long elapsedSeconds = elapsedTimeMillis / 1000;
                        long minutes = elapsedSeconds / 60;
                        long seconds = elapsedSeconds % 60;

                        System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
                        TimeUnit.SECONDS.sleep(5);
                    }
                }
            }

            long elapsedTimeMillis = System.currentTimeMillis() - startTime;
            long elapsedSeconds = elapsedTimeMillis / 1000;
            long minutes = elapsedSeconds / 60;
            long seconds = elapsedSeconds % 60;

            System.out.println(String.format("Cluster is available! Total Elapsed
Time: %02d:%02d", minutes, seconds));

        } catch (RedshiftException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void createTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName) {
        try {
            ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
                .clusterIdentifier(clusterId)
```

```
                .dbUser(userName)
                .database(databaseName)
                .sql("CREATE TABLE Movies ("
                    + "id INT PRIMARY KEY, "
                    + "title VARCHAR(100), "
                    + "year INT)")
                .build();

            redshiftDataClient.executeStatement(createTableRequest);
            System.out.println("Table created: Movies");

        } catch (RedshiftDataException e) {
            System.err.println("Error creating table: " + e.getMessage());
            System.exit(1);
        }
    }

    public static void createCluster(RedshiftClient redshiftClient, String
 clusterId, String masterUsername,
                                     String masterUserPassword) {
        try {
            CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
                .clusterIdentifier(clusterId)
                .masterUsername(masterUsername)
                .masterUserPassword(masterUserPassword)
                .nodeType("ra3.4xlarge")
                .publiclyAccessible(true)
                .numberOfNodes(2)
                .build();

            CreateClusterResponse clusterResponse =
 redshiftClient.createCluster(clusterRequest);
            System.out.println("Created cluster " +
 clusterResponse.cluster().clusterIdentifier());

        } catch (RedshiftException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [createCluster]
  - [describeClusters]
  - [describeStatement]
  - [executeStatement]
  - [getStatementResult]
  - [listDatabasesPaginator]
  - [modifyCluster]

Amazon Rekognition examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Rekognition.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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]
- [Scenarios]

Actions

`CompareFaces`

The following code example shows how to use `CompareFaces`.

For more information, see [Comparing faces in images].

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <pathSource> <pathTarget>

                Where:
                   pathSource - The path to the source image (for example, C:\\AWS\
\pic1.png).\s
                    pathTarget - The path to the target image (for example, C:\\AWS\
\pic2.png).\s
```

```
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        Float similarityThreshold = 70F;
        String sourceImage = args[0];
        String targetImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        compareTwoFaces(rekClient, similarityThreshold, sourceImage, targetImage);
        rekClient.close();
    }

    public static void compareTwoFaces(RekognitionClient rekClient, Float
 similarityThreshold, String sourceImage,
            String targetImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            InputStream tarStream = new FileInputStream(targetImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            Image tarImage = Image.builder()
                    .bytes(targetBytes)
                    .build();

            CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                    .sourceImage(souImage)
                    .targetImage(tarImage)
                    .similarityThreshold(similarityThreshold)
                    .build();

            // Compare the two images.
```

```
            CompareFacesResponse compareFacesResult =
 rekClient.compareFaces(facesRequest);
            List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();
            for (CompareFacesMatch match : faceDetails) {
                ComparedFace face = match.face();
                BoundingBox position = face.boundingBox();
                System.out.println("Face at " + position.left().toString()
                        + " " + position.top()
                        + " matches with " + face.confidence().toString()
                        + "% confidence.");

            }
            List<ComparedFace> uncompared = compareFacesResult.unmatchedFaces();
            System.out.println("There was " + uncompared.size() + " face(s) that did
 not match");
            System.out.println("Source image rotation: " +
 compareFacesResult.sourceImageOrientationCorrection());
            System.out.println("target image rotation: " +
 compareFacesResult.targetImageOrientationCorrection());

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println("Failed to load source image " + sourceImage);
            System.exit(1);
        }
    }
}
```

- For API details, see CompareFaces in *AWS SDK for Java 2.x API Reference*.

CreateCollection

The following code example shows how to use `CreateCollection`.

For more information, see Creating a collection.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionName>\s

                Where:
                   collectionName - The name of the collection.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void createMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
```

```
            CreateCollectionRequest collectionRequest =
 CreateCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            CreateCollectionResponse collectionResponse =
 rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " +
 collectionResponse.collectionArn());
            System.out.println("Status code: " +
 collectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see CreateCollection in *AWS SDK for Java 2.x API Reference*.

DeleteCollection

The following code example shows how to use `DeleteCollection`.

For more information, see Deleting a collection.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId>\s

                Where:
                   collectionId - The id of the collection to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void deleteMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
 DeleteCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            DeleteCollectionResponse deleteCollectionResponse =
 rekClient.deleteCollection(deleteCollectionRequest);
```

```
            System.out.println(collectionId + ": " +
 deleteCollectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteCollection in *AWS SDK for Java 2.x API Reference*.

DeleteFaces

The following code example shows how to use DeleteFaces.

For more information, see Deleting faces from a collection.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:    <collectionId> <faceId>\s

                Where:
                   collectionId - The id of the collection from which faces are
 deleted.\s

                   faceId - The id of the face to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    public static void deleteFacesCollection(RekognitionClient rekClient,
            String collectionId,
            String faceId) {

        try {
            DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceIds(faceId)
                    .build();

            rekClient.deleteFaces(deleteFacesRequest);
            System.out.println("The face was deleted from the collection.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
}
```

- For API details, see DeleteFaces in *AWS SDK for Java 2.x API Reference*.

DescribeCollection

The following code example shows how to use `DescribeCollection`.

For more information, see Describing a collection.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionName>
```

```
                Where:
                   collectionName - The name of the Amazon Rekognition collection.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    public static void describeColl(RekognitionClient rekClient, String
 collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
 DescribeCollectionRequest.builder()
                    .collectionId(collectionName)
                    .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                    .describeCollection(describeCollectionRequest);
            System.out.println("Collection Arn : " +
 describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
 describeCollectionResponse.creationTimestamp().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DescribeCollection in *AWS SDK for Java 2.x API Reference*.

DetectFaces

The following code example shows how to use `DetectFaces`.

For more information, see [Detecting faces in an image](#).

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <sourceImage>
```

```
                Where:
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectFacesinImage(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectFacesinImage(RekognitionClient rekClient, String
 sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                    .attributes(Attribute.ALL)
                    .image(souImage)
                    .build();

            DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
            List<FaceDetail> faceDetails = facesResponse.faceDetails();
            for (FaceDetail face : faceDetails) {
                AgeRange ageRange = face.ageRange();
                System.out.println("The detected face is estimated to be between "
                        + ageRange.low().toString() + " and " +
 ageRange.high().toString()
                        + " years old.");
```

```
                System.out.println("There is a smile : " +
 face.smile().value().toString());
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [DetectFaces](#) in *AWS SDK for Java 2.x API Reference*.

`DetectLabels`

The following code example shows how to use `DetectLabels`.

For more information, see [Detecting labels in an image](#).

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <sourceImage>

                Where:
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectImageLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectImageLabels(RekognitionClient rekClient, String
 sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();
```

```
            DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
                    .image(souImage)
                    .maxLabels(10)
                    .build();

            DetectLabelsResponse labelsResponse =
 rekClient.detectLabels(detectLabelsRequest);
            List<Label> labels = labelsResponse.labels();
            System.out.println("Detected labels for the given photo");
            for (Label label : labels) {
                System.out.println(label.name() + ": " +
 label.confidence().toString());
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [DetectLabels](#) in *AWS SDK for Java 2.x API Reference*.

`DetectModerationLabels`

The following code example shows how to use `DetectModerationLabels`.

For more information, see [Detecting inappropriate images](#).

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <sourceImage>

                Where:
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectModLabels(rekClient, sourceImage);
```

```
        rekClient.close();
    }

    public static void detectModLabels(RekognitionClient rekClient, String
 sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
 DetectModerationLabelsRequest.builder()
                    .image(souImage)
                    .minConfidence(60F)
                    .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                    .detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels =
 moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");
            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                        + "\n Confidence: " + label.confidence().toString() + "%"
                        + "\n Parent:" + label.parentName());
            }

        } catch (RekognitionException | FileNotFoundException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- For API details, see DetectModerationLabels in *AWS SDK for Java 2.x API Reference*.

DetectText

The following code example shows how to use `DetectText`.

For more information, see [Detecting text in an image](#).

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DetectTextRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectTextResponse;
import software.amazon.awssdk.services.rekognition.model.TextDetection;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <sourceImage>

                Where:
                   sourceImage - The path to the image that contains text (for
 example, C:\\AWS\\pic1.png).\s
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectTextLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectTextLabels(RekognitionClient rekClient, String
 sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            DetectTextRequest textRequest = DetectTextRequest.builder()
                    .image(souImage)
                    .build();

            DetectTextResponse textResponse = rekClient.detectText(textRequest);
            List<TextDetection> textCollection = textResponse.textDetections();
            System.out.println("Detected lines and words");
            for (TextDetection text : textCollection) {
                System.out.println("Detected: " + text.detectedText());
                System.out.println("Confidence: " + text.confidence().toString());
                System.out.println("Id : " + text.id());
                System.out.println("Parent Id: " + text.parentId());
                System.out.println("Type: " + text.type());
                System.out.println();
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

- For API details, see DetectText in *AWS SDK for Java 2.x API Reference*.

IndexFaces

The following code example shows how to use `IndexFaces`.

For more information, see Adding faces to a collection.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.IndexFacesResponse;
import software.amazon.awssdk.services.rekognition.model.IndexFacesRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.QualityFilter;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceRecord;
import software.amazon.awssdk.services.rekognition.model.UnindexedFace;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Reason;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddFacesToCollection {
    public static void main(String[] args) {

        final String usage = """

                Usage:     <collectionId> <sourceImage>

                Where:
                    collectionName - The name of the collection.
                    sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void addToCollection(RekognitionClient rekClient, String
 collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                    .collectionId(collectionId)
                    .image(souImage)
```

```
                    .maxFaces(1)
                    .qualityFilter(QualityFilter.AUTO)
                    .detectionAttributes(Attribute.DEFAULT)
                    .build();

            IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
            System.out.println("Results for the image");
            System.out.println("\n Faces indexed:");
            List<FaceRecord> faceRecords = facesResponse.faceRecords();
            for (FaceRecord faceRecord : faceRecords) {
                System.out.println("  Face ID: " + faceRecord.face().faceId());
                System.out.println("  Location:" +
 faceRecord.faceDetail().boundingBox().toString());
            }

            List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
            System.out.println("Faces not indexed:");
            for (UnindexedFace unindexedFace : unindexedFaces) {
                System.out.println("  Location:" +
 unindexedFace.faceDetail().boundingBox().toString());
                System.out.println("  Reasons:");
                for (Reason reason : unindexedFace.reasons()) {
                    System.out.println("Reason:  " + reason);
                }
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see IndexFaces in *AWS SDK for Java 2.x API Reference*.

ListCollections

The following code example shows how to use `ListCollections`.

For more information, see Listing collections.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/rekognition/#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
 ListCollectionsRequest.builder()
                    .maxResults(10)
                    .build();
```

```
            ListCollectionsResponse response =
 rekClient.listCollections(listCollectionsRequest);
            List<String> collectionIds = response.collectionIds();
            for (String resultId : collectionIds) {
                System.out.println(resultId);
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListCollections in *AWS SDK for Java 2.x API Reference*.

ListFaces

The following code example shows how to use `ListFaces`.

For more information, see Listing faces in a collection.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId>

                Where:
                   collectionId - The name of the collection.\s
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void listFacesCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                    .collectionId(collectionId)
                    .maxResults(10)
                    .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face : faces) {
                System.out.println("Confidence level there is a face: " +
 face.confidence());
```

```
                System.out.println("The face Id value is " + face.faceId());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [ListFaces](#) in *AWS SDK for Java 2.x API Reference*.

RecognizeCelebrities

The following code example shows how to use `RecognizeCelebrities`.

For more information, see [Recognizing celebrities in an image](#).

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
import
 software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import
 software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Celebrity;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = """
                Usage:    <sourceImage>

                Where:
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, sourceImage);
        rekClient.close();
    }

    public static void recognizeAllCelebrities(RekognitionClient rekClient, String
 sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();
```

```
            RecognizeCelebritiesRequest request =
 RecognizeCelebritiesRequest.builder()
                    .image(souImage)
                    .build();

            RecognizeCelebritiesResponse result =
 rekClient.recognizeCelebrities(request);
            List<Celebrity> celebs = result.celebrityFaces();
            System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
            for (Celebrity celebrity : celebs) {
                System.out.println("Celebrity recognized: " + celebrity.name());
                System.out.println("Celebrity ID: " + celebrity.id());

                System.out.println("Further information (if available):");
                for (String url : celebrity.urls()) {
                    System.out.println(url);
                }
                System.out.println();
            }
            System.out.println(result.unrecognizedFaces().size() + " face(s) were
 unrecognized.");

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see RecognizeCelebrities in *AWS SDK for Java 2.x API Reference*.

SearchFaces

The following code example shows how to use `SearchFaces`.

For more information, see Searching for a face (face ID).

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s

                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void searchFaceInCollection(RekognitionClient rekClient, String
 collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceImage));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            SearchFacesByImageRequest facesByImageRequest =
 SearchFacesByImageRequest.builder()
                    .image(souImage)
                    .maxFaces(10)
                    .faceMatchThreshold(70F)
                    .collectionId(collectionId)
                    .build();

            SearchFacesByImageResponse imageResponse =
 rekClient.searchFacesByImage(facesByImageRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " + face.similarity());
                System.out.println();
            }
```

```
        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see SearchFaces in *AWS SDK for Java 2.x API Reference*.

SearchFacesByImage

The following code example shows how to use `SearchFacesByImage`.

For more information, see Searching for a face (image).

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFacebyId(rekClient, collectionId, faceId);
        rekClient.close();
    }

    public static void searchFacebyId(RekognitionClient rekClient, String
 collectionId, String faceId) {
        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceId(faceId)
                    .faceMatchThreshold(70F)
                    .maxFaces(2)
                    .build();

            SearchFacesResponse imageResponse =
 rekClient.searchFaces(searchFacesRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " + face.similarity());
                System.out.println();
```

```
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [SearchFacesByImage](#) in *AWS SDK for Java 2.x 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 Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get celebrity results from a video located in an Amazon S3 bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startCelebrityDetection(rekClient, channel, bucket, video);
        getCelebrityDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startCelebrityDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartCelebrityRecognitionRequest recognitionRequest =
 StartCelebrityRecognitionRequest.builder()
                    .jobTag("Celebrities")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();
```

```
            StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
 rekClient
                    .startCelebrityRecognition(recognitionRequest);
            startJobId = startCelebrityRecognitionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getCelebrityDetectionResults(RekognitionClient rekClient) {

        try {
            String paginationToken = null;
            GetCelebrityRecognitionResponse recognitionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (recognitionResponse != null)
                    paginationToken = recognitionResponse.nextToken();

                GetCelebrityRecognitionRequest recognitionRequest =
 GetCelebrityRecognitionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {
                    recognitionResponse =
 rekClient.getCelebrityRecognition(recognitionRequest);
                    status = recognitionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
```

```
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = recognitionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<CelebrityRecognition> celebs =
 recognitionResponse.celebrities();
                for (CelebrityRecognition celeb : celebs) {
                    long seconds = celeb.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    CelebrityDetail details = celeb.celebrity();
                    System.out.println("Name: " + details.name());
                    System.out.println("Id: " + details.id());
                    System.out.println();
                }

            } while (recognitionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detect labels in a video by a label detection operation.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of the video (for example, people.mp4).\s
                   queueUrl- The URL of a SQS queue.\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
```

```
                  roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
               """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startLabels(rekClient, channel, bucket, video);
        getLabelJob(rekClient, sqs, queueUrl);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();
```

```
            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
```

```
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
```

```
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
```

```
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels = labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
 detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
 instance.confidence().toString());
                            System.out.println("        Bounding box: " +
 instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
 ":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);
```

```
        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

Detect faces in a video stored in an Amazon S3 bucket.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of the video (for example, people.mp4).\s
                   queueUrl- The URL of a SQS queue.\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();
```

```
        startLabels(rekClient, channel, bucket, video);
        getLabelJob(rekClient, sqs, queueUrl);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
 StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
 rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
 GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();
```

```
                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
```

```
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;
```

```
        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels = labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
instance.confidence().toString());
                            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                        }
```

```
                    }
                    System.out.println("   Parent labels for " + label.name() +
 ":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

Detect inappropriate or offensive content in a video stored in an Amazon S3 bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
```

```
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startModerationDetection(rekClient, channel, bucket, video);
        getModResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startModerationDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {

        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartContentModerationRequest modDetectionRequest =
 StartContentModerationRequest.builder()
                    .jobTag("Moderation")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartContentModerationResponse startModDetectionResult = rekClient
                    .startContentModeration(modDetectionRequest);
            startJobId = startModDetectionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getModResults(RekognitionClient rekClient) {
        try {
```

```
            String paginationToken = null;
            GetContentModerationResponse modDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (modDetectionResponse != null)
                    paginationToken = modDetectionResponse.nextToken();

                GetContentModerationRequest modRequest = 
GetContentModerationRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    modDetectionResponse = 
rekClient.getContentModeration(modRequest);
                    status = modDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = modDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<ContentModerationDetection> mods = 
modDetectionResponse.moderationLabels();
```

```
                for (ContentModerationDetection mod : mods) {
                    long seconds = mod.timestamp() / 1000;
                    System.out.print("Mod label: " + seconds + " ");
                    System.out.println(mod.moderationLabel().toString());
                    System.out.println();
                }

            } while (modDetectionResponse != null &&
 modDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detect technical cue segments and shot detection segments in a video stored in an Amazon S3 bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
```

```
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectSegment {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
```

```
        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startSegmentDetection(rekClient, channel, bucket, video);
        getSegmentResults(rekClient);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startSegmentDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartShotDetectionFilter cueDetectionFilter =
 StartShotDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
 StartTechnicalCueDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartSegmentDetectionFilters filters =
 StartSegmentDetectionFilters.builder()
                    .shotFilter(cueDetectionFilter)
```

```
                    .technicalCueFilter(technicalCueDetectionFilter)
                    .build();

            StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
                    .video(vidOb)
                    .filters(filters)
                    .build();

            StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
            startJobId = segDetectionResponse.jobId();

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getSegmentResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetSegmentDetectionResponse segDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (segDetectionResponse != null)
                    paginationToken = segDetectionResponse.nextToken();

                GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
```

```
                    segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                    status = segDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }
                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
                for (VideoMetadata metaData : videoMetaData) {
                    System.out.println("Format: " + metaData.format());
                    System.out.println("Codec: " + metaData.codec());
                    System.out.println("Duration: " + metaData.durationMillis());
                    System.out.println("FrameRate: " + metaData.frameRate());
                    System.out.println("Job");
                }

                List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
                for (SegmentDetection detectedSegment : detectedSegments) {
                    String type = detectedSegment.type().toString();
                    if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                        System.out.println("Technical Cue");
                        TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
                        System.out.println("\tType: " + segmentCue.type());
                        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
                    }

                    if (type.contains(SegmentType.SHOT.toString())) {
                        System.out.println("Shot");
                        ShotSegment segmentShot = detectedSegment.shotSegment();
                        System.out.println("\tIndex " + segmentShot.index());
                        System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
```

```
                    }

                    long seconds = detectedSegment.durationMillis();
                    System.out.println("\tDuration : " + seconds + " milliseconds");
                    System.out.println("\tStart time code: " +
 detectedSegment.startTimecodeSMPTE());
                    System.out.println("\tEnd time code: " +
 detectedSegment.endTimecodeSMPTE());
                    System.out.println("\tDuration time code: " +
 detectedSegment.durationSMPTE());
                    System.out.println();
                }

            } while (segDetectionResponse != null &&
 segDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detect text in a video stored in a video stored in an Amazon S3 bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startTextLabels(rekClient, channel, bucket, video);
```

```
        getTextResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startTextLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartTextDetectionRequest labelDetectionRequest =
 StartTextDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartTextDetectionResponse labelDetectionResponse =
 rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getTextResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetTextDetectionResponse textDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;
```

```
            do {
                if (textDetectionResponse != null)
                    paginationToken = textDetectionResponse.nextToken();

                GetTextDetectionRequest recognitionRequest =
 GetTextDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    textDetectionResponse =
 rekClient.getTextDetection(recognitionRequest);
                    status = textDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = textDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<TextDetectionResult> labels =
 textDetectionResponse.textDetections();
                for (TextDetectionResult detectedText : labels) {
                    System.out.println("Confidence: " +
 detectedText.textDetection().confidence().toString());
                    System.out.println("Id : " + detectedText.textDetection().id());
                    System.out.println("Parent Id: " +
 detectedText.textDetection().parentId());
```

```
                    System.out.println("Type: " +
 detectedText.textDetection().type());
                    System.out.println("Text: " +
 detectedText.textDetection().detectedText());
                    System.out.println();
                }

            } while (textDetectionResponse != null &&
 textDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detect people in a video stored in a video stored in an Amazon S3 bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class VideoPersonDetection {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }
```

```
    public static void startPersonLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartPersonTrackingRequest personTrackingRequest =
 StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidOb)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
 rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPersonDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
                    paginationToken = personTrackingResult.nextToken();
```

```
                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<PersonDetection> detectedPersons =
personTrackingResult.persons();
                for (PersonDetection detectedPerson : detectedPersons) {
                    long seconds = detectedPerson.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
detectedPerson.person().index());
                    System.out.println();
                }
```

```
            } while (personTrackingResult != null &&
 personTrackingResult.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

Route 53 domain registration examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Route 53 domain registration.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Route 53 domain registration

The following code examples show how to get started using Route 53 domain registration.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/route53#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.route53domains.Route53DomainsClient;
import software.amazon.awssdk.services.route53.model.Route53Exception;
import software.amazon.awssdk.services.route53domains.model.DomainPrice;
import software.amazon.awssdk.services.route53domains.model.ListPricesRequest;
import software.amazon.awssdk.services.route53domains.model.ListPricesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code examples performs the following operation:
 *
 * 1. Invokes ListPrices for at least one domain type, such as the “com” type
 * and displays the prices for Registration and Renewal.
 *
 */
public class HelloRoute53 {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
```

```
        final String usage = "\n" +
                "Usage:\n" +
                "    <hostedZoneId> \n\n" +
                "Where:\n" +
                "    hostedZoneId - The id value of an existing hosted zone. \n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Invokes ListPrices for at least one domain type.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
 domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .maxItems(10)
                    .tld(domainType)
                    .build();

            ListPricesResponse response =
 route53DomainsClient.listPrices(pricesRequest);
            List<DomainPrice> prices = response.prices();
            for (DomainPrice pr : prices) {
                System.out.println("Name: " + pr.name());
                System.out.println(
                        "Registration: " + pr.registrationPrice().price() + " " +
 pr.registrationPrice().currency());
                System.out.println("Renewal: " + pr.renewalPrice().price() + " " +
 pr.renewalPrice().currency());
                System.out.println("Transfer: " + pr.transferPrice().price() + " " +
 pr.transferPrice().currency());
```

```
                System.out.println("Transfer: " + pr.transferPrice().price() + " " +
 pr.transferPrice().currency());
                System.out.println("Change Ownership: " +
 pr.changeOwnershipPrice().price() + " "
                        + pr.changeOwnershipPrice().currency());
                System.out.println(
                        "Restoration: " + pr.restorationPrice().price() + " " +
 pr.restorationPrice().currency());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListPrices in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

CheckDomainAvailability

The following code example shows how to use `CheckDomainAvailability`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainAvailabilityResponse response = route53DomainsClient
                    .checkDomainAvailability(availabilityRequest);
            System.out.println(domainSuggestion + " is " +
response.availability().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see [CheckDomainAvailability](#) in *AWS SDK for Java 2.x API Reference*.

CheckDomainTransferability

The following code example shows how to use `CheckDomainTransferability`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();
```

```
            CheckDomainTransferabilityResponse response = route53DomainsClient
                    .checkDomainTransferability(transferabilityRequest);
            System.out.println("Transferability: " +
response.transferability().transferable().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see CheckDomainTransferability in *AWS SDK for Java 2.x API Reference*.

`GetDomainDetail`

The following code example shows how to use `GetDomainDetail`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
response.registrantContact().firstName());
            System.out.println("The contact last name is " +
response.registrantContact().lastName());
            System.out.println("The contact org name is " +
response.registrantContact().organizationName());
```

```
        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see [GetDomainDetail](#) in *AWS SDK for Java 2.x API Reference*.

GetDomainSuggestions

The following code example shows how to use `GetDomainSuggestions`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void listDomainSuggestions(Route53DomainsClient
 route53DomainsClient, String domainSuggestion) {
        try {
            GetDomainSuggestionsRequest suggestionsRequest =
 GetDomainSuggestionsRequest.builder()
                    .domainName(domainSuggestion)
                    .suggestionCount(5)
                    .onlyAvailable(true)
                    .build();

            GetDomainSuggestionsResponse response =
 route53DomainsClient.getDomainSuggestions(suggestionsRequest);
            List<DomainSuggestion> suggestions = response.suggestionsList();
            for (DomainSuggestion suggestion : suggestions) {
                System.out.println("Suggestion Name: " + suggestion.domainName());
                System.out.println("Availability: " + suggestion.availability());
                System.out.println(" ");
            }
```

```
        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see GetDomainSuggestions in *AWS SDK for Java 2.x API Reference*.

GetOperationDetail

The following code example shows how to use `GetOperationDetail`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                    .operationId(operationId)
                    .build();

            GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is " + response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see GetOperationDetail in *AWS SDK for Java 2.x API Reference*.

`ListDomains`

The following code example shows how to use `ListDomains`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
            ListDomainsIterable listRes =
 route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                    .flatMap(r -> r.domains().stream())
                    .forEach(content -> System.out.println("The domain name is " +
 content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see [ListDomains](#) in *AWS SDK for Java 2.x API Reference*.

`ListOperations`

The following code example shows how to use `ListOperations`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void listOperations(Route53DomainsClient route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            localDateTime = localDateTime.minusYears(1);
            Instant myTime = localDateTime.toInstant(zoneOffset);

            ListOperationsRequest operationsRequest =
 ListOperationsRequest.builder()
                    .submittedSince(myTime)
                    .build();

            ListOperationsIterable listRes =
 route53DomainsClient.listOperationsPaginator(operationsRequest);
            listRes.stream()
                    .flatMap(r -> r.operations().stream())
                    .forEach(content -> System.out.println(" Operation Id: " +
 content.operationId() +
                            " Status: " + content.statusAsString() +
                            " Date: " + content.submittedDate()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListOperations in *AWS SDK for Java 2.x API Reference*.

ListPrices

The following code example shows how to use `ListPrices`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listPrices(Route53DomainsClient route53DomainsClient, String
 domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .tld(domainType)
                    .build();

            ListPricesIterable listRes =
 route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
                    .flatMap(r -> r.prices().stream())
                    .forEach(content -> System.out.println(" Name: " +
 content.name() +
                            " Registration: " + content.registrationPrice().price()
 + " "
                            + content.registrationPrice().currency() +
                            " Renewal: " + content.renewalPrice().price() + " " +
 content.renewalPrice().currency()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListPrices in *AWS SDK for Java 2.x API Reference*.

RegisterDomain

The following code example shows how to use `RegisterDomain`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/route53#code-examples).

```
    public static String requestDomainRegistration(Route53DomainsClient
 route53DomainsClient,
            String domainSuggestion,
            String phoneNumber,
            String email,
            String firstName,
            String lastName,
            String city) {

        try {
            ContactDetail contactDetail = ContactDetail.builder()
                    .contactType(ContactType.COMPANY)
                    .state("LA")
                    .countryCode(CountryCode.IN)
                    .email(email)
                    .firstName(firstName)
                    .lastName(lastName)
                    .city(city)
                    .phoneNumber(phoneNumber)
                    .organizationName("My Org")
                    .addressLine1("My Address")
                    .zipCode("123 123")
                    .build();

            RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                    .adminContact(contactDetail)
                    .registrantContact(contactDetail)
                    .techContact(contactDetail)
                    .domainName(domainSuggestion)
                    .autoRenew(true)
                    .durationInYears(1)
                    .build();
```

```
            RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
            System.out.println("Registration requested. Operation Id: " +
response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see RegisterDomain in *AWS SDK for Java 2.x API Reference*.

`ViewBilling`

The following code example shows how to use `ViewBilling`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            LocalDateTime localDateTime2 = localDateTime.minusYears(1);
            Instant myStartTime = localDateTime2.toInstant(zoneOffset);
            Instant myEndTime = localDateTime.toInstant(zoneOffset);

            ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                    .start(myStartTime)
```

```
                    .end(myEndTime)
                    .build();

            ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
            listRes.stream()
                    .flatMap(r -> r.billingRecords().stream())
                    .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                            " Operation: " + content.operationAsString() +
                            " Price: " + content.price()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ViewBilling in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with domains

The following code example shows how to:

- List current domains, and list operations in the past year.
- View billing for the past year, and view prices for domain types.
- Get domain suggestions.
- Check domain availability and transferability.
- Optionally, request a domain registration.
- Get an operation detail.
- Optionally, get a domain detail.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/route53#code-examples).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example uses pagination methods where applicable. For example, to list
 * domains, the
 * listDomainsPaginator method is used. For more information about pagination,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/pagination.html
 *
 * This Java code example performs the following operations:
 *
 * 1. List current domains.
 * 2. List operations in the past year.
 * 3. View billing for the account in the past year.
 * 4. View prices for domain types.
 * 5. Get domain suggestions.
 * 6. Check domain availability.
 * 7. Check domain transferability.
 * 8. Request a domain registration.
 * 9. Get operation details.
 * 10. Optionally, get domain details.
 */

public class Route53Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <domainType> <phoneNumber> <email> <domainSuggestion>
<firstName> <lastName> <city>

                Where:
                    domainType - The domain type (for example, com).\s
                    phoneNumber - The phone number to use (for example,
+91.9966564xxx)      email - The email address to use.      domainSuggestion - The
domain suggestion (for example, findmy.accountants).\s
                    firstName - The first name to use to register a domain.\s
                    lastName -  The last name to use to register a domain.\s
                    city - the city to use to register a domain.\s
                    """;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        String phoneNumber = args[1];
        String email = args[2];
        String domainSuggestion = args[3];
        String firstName = args[4];
        String lastName = args[5];
        String city = args[6];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Route 53 domains example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. List current domains.");
        listDomains(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List operations in the past year.");
```

```
        listOperations(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. View billing for the account in the past year.");
        listBillingRecords(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. View prices for domain types.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get domain suggestions.");
        listDomainSuggestions(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Check domain availability.");
        checkDomainAvailability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Check domain transferability.");
        checkDomainTransferability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Request a domain registration.");
        String opId = requestDomainRegistration(route53DomainsClient,
 domainSuggestion, phoneNumber, email, firstName,
                lastName, city);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get operation details.");
        getOperationalDetail(route53DomainsClient, opId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get domain details.");
        System.out.println("Note: You must have a registered domain to get
 details.");
```

```
        System.out.println("Otherwise, an exception is thrown that states ");
        System.out.println("Domain xxxxxxx not found in xxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
 String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            GetDomainDetailResponse response =
 route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
 response.registrantContact().firstName());
            System.out.println("The contact last name is " +
 response.registrantContact().lastName());
            System.out.println("The contact org name is " +
 response.registrantContact().organizationName());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getOperationalDetail(Route53DomainsClient
 route53DomainsClient, String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
 GetOperationDetailRequest.builder()
                    .operationId(operationId)
                    .build();

            GetOperationDetailResponse response =
 route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is " + response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }

    public static String requestDomainRegistration(Route53DomainsClient
 route53DomainsClient,
            String domainSuggestion,
            String phoneNumber,
            String email,
            String firstName,
            String lastName,
            String city) {

        try {
            ContactDetail contactDetail = ContactDetail.builder()
                    .contactType(ContactType.COMPANY)
                    .state("LA")
                    .countryCode(CountryCode.IN)
                    .email(email)
                    .firstName(firstName)
                    .lastName(lastName)
                    .city(city)
                    .phoneNumber(phoneNumber)
                    .organizationName("My Org")
                    .addressLine1("My Address")
                    .zipCode("123 123")
                    .build();

            RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                    .adminContact(contactDetail)
                    .registrantContact(contactDetail)
                    .techContact(contactDetail)
                    .domainName(domainSuggestion)
                    .autoRenew(true)
                    .durationInYears(1)
                    .build();

            RegisterDomainResponse response =
 route53DomainsClient.registerDomain(domainRequest);
            System.out.println("Registration requested. Operation Id: " +
 response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
        return "";
    }

    public static void checkDomainTransferability(Route53DomainsClient
 route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainTransferabilityRequest transferabilityRequest =
 CheckDomainTransferabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainTransferabilityResponse response = route53DomainsClient
                    .checkDomainTransferability(transferabilityRequest);
            System.out.println("Transferability: " +
 response.transferability().transferable().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void checkDomainAvailability(Route53DomainsClient
 route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainAvailabilityRequest availabilityRequest =
 CheckDomainAvailabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainAvailabilityResponse response = route53DomainsClient
                    .checkDomainAvailability(availabilityRequest);
            System.out.println(domainSuggestion + " is " +
 response.availability().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomainSuggestions(Route53DomainsClient
 route53DomainsClient, String domainSuggestion) {
```

```
        try {
            GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                    .domainName(domainSuggestion)
                    .suggestionCount(5)
                    .onlyAvailable(true)
                    .build();

            GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
            List<DomainSuggestion> suggestions = response.suggestionsList();
            for (DomainSuggestion suggestion : suggestions) {
                System.out.println("Suggestion Name: " + suggestion.domainName());
                System.out.println("Availability: " + suggestion.availability());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .tld(domainType)
                    .build();

            ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
                    .flatMap(r -> r.prices().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.name() +
                            " Registration: " + content.registrationPrice().price()
+ " "
                            + content.registrationPrice().currency() +
                            " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
```

```
            System.exit(1);
        }
    }

    public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
 {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            LocalDateTime localDateTime2 = localDateTime.minusYears(1);
            Instant myStartTime = localDateTime2.toInstant(zoneOffset);
            Instant myEndTime = localDateTime.toInstant(zoneOffset);

            ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                    .start(myStartTime)
                    .end(myEndTime)
                    .build();

            ViewBillingIterable listRes =
 route53DomainsClient.viewBillingPaginator(viewBillingRequest);
            listRes.stream()
                    .flatMap(r -> r.billingRecords().stream())
                    .forEach(content -> System.out.println(" Bill Date:: " +
 content.billDate() +
                            " Operation: " + content.operationAsString() +
                            " Price: " + content.price()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listOperations(Route53DomainsClient route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            localDateTime = localDateTime.minusYears(1);
            Instant myTime = localDateTime.toInstant(zoneOffset);
```

```
            ListOperationsRequest operationsRequest =
 ListOperationsRequest.builder()
                    .submittedSince(myTime)
                    .build();

            ListOperationsIterable listRes =
 route53DomainsClient.listOperationsPaginator(operationsRequest);
            listRes.stream()
                    .flatMap(r -> r.operations().stream())
                    .forEach(content -> System.out.println(" Operation Id: " +
 content.operationId() +
                            " Status: " + content.statusAsString() +
                            " Date: " + content.submittedDate()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
            ListDomainsIterable listRes =
 route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                    .flatMap(r -> r.domains().stream())
                    .forEach(content -> System.out.println("The domain name is " +
 content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions

- GetOperationDetail
- ListDomains
- ListOperations
- ListPrices
- RegisterDomain
- ViewBilling

Amazon S3 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon S3.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon S3

The following code examples show how to get started using Amazon S3.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListBuckets in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios
- Serverless examples

Actions

CopyObject

The following code example shows how to use `CopyObject`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Copy an object using an S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <objectKey> <fromBucket> <toBucket>
```

```
                Where:
                    objectKey - The name of the object (for example, book.pdf).
                    fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
                    toBucket - The S3 bucket to copy the object to (for example,
bucket2).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String objectKey = args[0];
        String fromBucket = args[1];
        String toBucket = args[2];
        System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        copyBucketObject(s3, fromBucket, objectKey, toBucket);
        s3.close();
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
                .sourceBucket(fromBucket)
                .sourceKey(objectKey)
                .destinationBucket(toBucket)
                .destinationKey(objectKey)
                .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
        return "";
    }
}
```

Use an S3TransferManager to copy an object from one bucket to another. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String bucketName,
            String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
                .sourceBucket(bucketName)
                .sourceKey(key)
                .destinationBucket(destinationBucket)
                .destinationKey(destinationKey)
                .build();

        CopyRequest copyRequest = CopyRequest.builder()
                .copyObjectRequest(copyObjectRequest)
                .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

- For API details, see CopyObject in *AWS SDK for Java 2.x API Reference*.

CreateBucket

The following code example shows how to use `CreateBucket`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/s3#code-examples).

Create a bucket.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the bucket to create. The bucket name
 must be unique, or an error occurs.
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Creating a bucket named %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        createBucket(s3, bucketName);
        s3.close();
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
 s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Create a bucket with object lock enabled.

```
    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
 bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }
```

- For API details, see CreateBucket in *AWS SDK for Java 2.x API Reference*.

`DeleteBucket`

The following code example shows how to use `DeleteBucket`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucket)
                .build();

        s3.deleteBucket(deleteBucketRequest);
```

```
        s3.close();
```

- For API details, see DeleteBucket in *AWS SDK for Java 2.x API Reference*.

DeleteBucketPolicy

The following code example shows how to use `DeleteBucketPolicy`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the policy from (for
 example, bucket1).""";
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    // Delete the bucket policy.
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteBucketPolicy in *AWS SDK for Java 2.x API Reference*.

`DeleteBucketWebsite`

The following code example shows how to use `DeleteBucketWebsite`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the website
 configuration from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket: %s
\n", bucketName);
        Region region = Region.US_EAST_1;
```

```
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteBucketWebsite in *AWS SDK for Java 2.x API Reference*.

`DeleteObjects`

The following code example shows how to use `DeleteObjects`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucketName>

                Where:
                   bucketName - the Amazon S3 bucket name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
```

```
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;

        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                    .key(keyName)
                    .build();

            putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
        }

        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
                .objects(keys)
                .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
 DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(del)
                    .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteObjects in *AWS SDK for Java 2.x API Reference*.

GetBucketAcl

The following code example shows how to use `GetBucketAcl`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <objectKey>

                Where:
                  bucketName - The Amazon S3 bucket to get the access control list
 (ACL) for.
                  objectKey - The object to get the ACL for.\s
```

```
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    public static String getBucketACL(S3Client s3, String objectKey, String 
bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
                System.out.format("  %s: %s\n", grant.grantee().id(), 
grant.permission());
                grantee = grant.grantee().id();
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return "";
    }
}
```

- For API details, see GetBucketAcl in *AWS SDK for Java 2.x API Reference*.

GetBucketPolicy

The following code example shows how to use `GetBucketPolicy`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>
```

```
                Where:
                    bucketName - The Amazon S3 bucket to get the policy from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- For API details, see GetBucketPolicy in *AWS SDK for Java 2.x API Reference*.

`GetObject`

The following code example shows how to use `GetObject`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Read data as a byte array using an S3Client.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <path>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
```

```
                    keyName - The key name.\s
                    path - The path where the file is written to.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getObjectBytes(s3, bucketName, keyName, path);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
 keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

Use an S3TransferManager to download an object in an S3 bucket to a local file. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
 transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
 downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
 downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Read tags that belong to an object using an S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
```

```
                .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    public static void listTags(S3Client s3, String bucketName, String keyName) {
        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet = tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
                System.out.println(tag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Get a URL for an object using an S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for  " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

Get an object by using the S3Presigner client object using an S3Client.

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents a text file.\s
                """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
```

```
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
                .region(region)
                .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
 String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            GetObjectPresignRequest getObjectPresignRequest =
 GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(60))
                    .getObjectRequest(getObjectRequest)
                    .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
 presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
 presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
 values) -> {
                values.forEach(value -> {
                    connection.addRequestProperty(header, value);
                });
            });

            // Send any request payload that the service needs (not needed when
            // isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);
```

```
                try (InputStream signedPayload =
 presignedGetObjectRequest.signedPayload().get().asInputStream();
                        OutputStream httpOutputStream =
 connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }

            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                System.out.println("Service returned response: ");
                IoUtils.copy(content, System.out);
            }

        } catch (S3Exception | IOException e) {
            e.getStackTrace();
        }
    }
}
```

Get an object by using a ResponseTransformer object and [S3Client](https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/s3/S3Client.html).

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */

public class GetDataResponseTransformer {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <path>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                    path - The path where the file is written to.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
 keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();
```

```
            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see GetObject in *AWS SDK for Java 2.x API Reference*.

`GetObjectLegalHold`

The following code example shows how to use `GetObjectLegalHold`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
 objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
 GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();
```

```
            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
"'");
        }

        return null;
    }
```

- For API details, see [GetObjectLegalHold](#) in *AWS SDK for Java 2.x API Reference*.

`GetObjectLockConfiguration`

The following code example shows how to use `GetObjectLockConfiguration`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
"+response.objectLockConfiguration().objectLockEnabled());
```

```
        System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
    }
```

- For API details, see GetObjectLockConfiguration in *AWS SDK for Java 2.x API Reference*.

`GetObjectRetention`

The following code example shows how to use `GetObjectRetention`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }
```

- For API details, see GetObjectRetention in *AWS SDK for Java 2.x API Reference*.

HeadObject

The following code example shows how to use `HeadObject`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Determine the content type of an object.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    public static void getContentType(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Get the restore status of an object.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    public static void checkStatus(S3Client s3, String bucketName, String keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
 response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

- For API details, see HeadObject in *AWS SDK for Java 2.x API Reference*.

`ListBuckets`

The following code example shows how to use `ListBuckets`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);

    }
```

```
    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
        List<Bucket> bucketList = response.buckets();
        for (Bucket bucket: bucketList) {
            System.out.println("Bucket name "+bucket.name());
        }
    }
}
```

- For API details, see ListBuckets in *AWS SDK for Java 2.x API Reference*.

`ListMultipartUploads`

The following code example shows how to use `ListMultipartUploads`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListMultipartUploads {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key() +
 "\", id = " + upload.uploadId());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

- For API details, see ListMultipartUploads in *AWS SDK for Java 2.x API Reference*.

ListObjectsV2

The following code example shows how to use `ListObjectsV2`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s
```

```
                Where:
                    bucketName - The Amazon S3 bucket from which objects are read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                System.out.print("\n The object is " + calKb(myValue.size()) + "
 KBs");
                System.out.print("\n The owner is " + myValue.owner());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // convert bytes to kbs.
    private static long calKb(Long val) {
        return val / 1024;
```

```
    }
}
```

List objects using pagination.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket from which objects are read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                    .bucket(bucketName)
                    .maxKeys(1)
```

```
                    .build();

            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                    .flatMap(r -> r.contents().stream())
                    .forEach(content -> System.out.println(" Key: " + content.key()
 + " size = " + content.size()));

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListObjectsV2 in *AWS SDK for Java 2.x API Reference*.

`PutBucketAcl`

The following code example shows how to use `PutBucketAcl`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <id>\s

                Where:
                  bucketName - The Amazon S3 bucket to grant permissions on.\s
                  id - The ID of the owner of this bucket (you can get this value
 from the AWS Management Console).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                    .grantee(builder -> builder.id(id)
```

```
                                .type(Type.CANONICAL_USER))
                        .permission(Permission.FULL_CONTROL)
                        .build();

                List<Grant> grantList2 = new ArrayList<>();
                grantList2.add(ownerGrant);

                AccessControlPolicy acl = AccessControlPolicy.builder()
                        .owner(builder -> builder.id(id))
                        .grants(grantList2)
                        .build();

                PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                        .bucket(bucketName)
                        .accessControlPolicy(acl)
                        .build();

                s3.putBucketAcl(putAclReq);

            } catch (S3Exception e) {
                e.printStackTrace();
                System.exit(1);
            }
        }
    }
```

- For API details, see PutBucketAcl in *AWS SDK for Java 2.x API Reference*.

PutBucketCors

The following code example shows how to use `PutBucketCors`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <accountId>\s

                Where:
                    bucketName - The Amazon S3 bucket to upload an object into.
                    accountId - The id of the account that owns the Amazon S3
 bucket.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
```

```
        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                    .bucket(bucketName)
                    .expectedBucketOwner(accountId)
                    .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
                    .bucket(bucketName)
                    .expectedBucketOwner(accountId)
                    .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.corsRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    public static void setCorsInformation(S3Client s3, String bucketName, String
 accountId) {
        List<String> allowMethods = new ArrayList<>();
        allowMethods.add("PUT");
        allowMethods.add("POST");
        allowMethods.add("DELETE");

        List<String> allowOrigins = new ArrayList<>();
        allowOrigins.add("http://example.com");
        try {
            // Define CORS rules.
            CORSRule corsRule = CORSRule.builder()
                    .allowedMethods(allowMethods)
                    .allowedOrigins(allowOrigins)
                    .build();

            List<CORSRule> corsRules = new ArrayList<>();
            corsRules.add(corsRule);
            CORSConfiguration configuration = CORSConfiguration.builder()
                    .corsRules(corsRules)
                    .build();

            PutBucketCorsRequest putBucketCorsRequest =
 PutBucketCorsRequest.builder()
                    .bucket(bucketName)
                    .corsConfiguration(configuration)
                    .expectedBucketOwner(accountId)
                    .build();

            s3.putBucketCors(putBucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see PutBucketCors in *AWS SDK for Java 2.x API Reference*.

PutBucketLifecycleConfiguration

The following code example shows how to use `PutBucketLifecycleConfiguration`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/s3#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class LifecycleConfiguration {
        public static void main(String[] args) {
                final String usage = """
```

```
                                Usage:
                                  <bucketName> <accountId>\s

                                Where:
                                  bucketName - The Amazon Simple Storage Service
(Amazon S3) bucket to upload an object into.
                                  accountId - The id of the account that owns the
Amazon S3 bucket.
                                """;

                if (args.length != 2) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String bucketName = args[0];
                String accountId = args[1];
                Region region = Region.US_EAST_1;
                S3Client s3 = S3Client.builder()
                                .region(region)
                                .build();

                setLifecycleConfig(s3, bucketName, accountId);
                getLifecycleConfig(s3, bucketName, accountId);
                deleteLifecycleConfig(s3, bucketName, accountId);
                System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
                s3.close();
        }

        public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
                try {
                        // Create a rule to archive objects with the
"glacierobjects/" prefix to Amazon
                        // S3 Glacier.
                        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                                        .prefix("glacierobjects/")
                                        .build();

                        Transition transition = Transition.builder()
```

```
.storageClass(TransitionStorageClass.GLACIER)
                                       .days(0)
                                       .build();

                       LifecycleRule rule1 = LifecycleRule.builder()
                                       .id("Archive immediately rule")
                                       .filter(ruleFilter)
                                       .transitions(transition)
                                       .status(ExpirationStatus.ENABLED)
                                       .build();

                       // Create a second rule.
                       Transition transition2 = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                       .days(0)
                                       .build();

                       List<Transition> transitionList = new ArrayList<>();
                       transitionList.add(transition2);

                       LifecycleRuleFilter ruleFilter2 =
LifecycleRuleFilter.builder()
                                       .prefix("glacierobjects/")
                                       .build();

                       LifecycleRule rule2 = LifecycleRule.builder()
                                       .id("Archive and then delete rule")
                                       .filter(ruleFilter2)
                                       .transitions(transitionList)
                                       .status(ExpirationStatus.ENABLED)
                                       .build();

                       // Add the LifecycleRule objects to an ArrayList.
                       ArrayList<LifecycleRule> ruleList = new ArrayList<>();
                       ruleList.add(rule1);
                       ruleList.add(rule2);

                       BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                                       .rules(ruleList)
                                       .build();
```

```
                        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                                        .builder()
                                        .bucket(bucketName)

.lifecycleConfiguration(lifecycleConfiguration)
                                        .expectedBucketOwner(accountId)
                                        .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }

        // Retrieve the configuration and add a new rule.
        public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
                try {
                        GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
                                        .builder()
                                        .bucket(bucketName)
                                        .expectedBucketOwner(accountId)
                                        .build();

                        GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
                        List<LifecycleRule> newList = new ArrayList<>();
                        List<LifecycleRule> rules = response.rules();
                        for (LifecycleRule rule : rules) {
                                newList.add(rule);
                        }

                        // Add a new rule with both a prefix predicate and a tag
predicate.
                        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                                        .prefix("YearlyDocuments/")
                                        .build();
```

```
                        Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(3650)
                                        .build();

                        LifecycleRule rule1 = LifecycleRule.builder()
                                        .id("NewRule")
                                        .filter(ruleFilter)
                                        .transitions(transition)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Add the new rule to the list.
                        newList.add(rule1);
                        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                                        .rules(newList)
                                        .build();

                        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                                        .builder()
                                        .bucket(bucketName)

.lifecycleConfiguration(lifecycleConfiguration)
                                        .expectedBucketOwner(accountId)
                                        .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }

        // Delete the configuration from the Amazon S3 bucket.
        public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
                try {
```

```
                        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
 DeleteBucketLifecycleRequest
                                        .builder()
                                        .bucket(bucketName)
                                        .expectedBucketOwner(accountId)
                                        .build();

                        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

- For API details, see PutBucketLifecycleConfiguration in *AWS SDK for Java 2.x API Reference*.

PutBucketNotificationConfiguration

The following code example shows how to use `PutBucketNotificationConfiguration`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Event;
import software.amazon.awssdk.services.s3.model.NotificationConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketNotificationConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.TopicConfiguration;
import java.util.ArrayList;
import java.util.List;
```

```
public class SetBucketEventBridgeNotification {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket.\s
                    topicArn - The Simple Notification Service topic ARN.\s
                    id - An id value used for the topic configuration. This value is
 displayed in the AWS Management Console.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String topicArn = args[1];
        String id = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        setBucketNotification(s3Client, bucketName, topicArn, id);
        s3Client.close();
    }

    public static void setBucketNotification(S3Client s3Client, String bucketName,
 String topicArn, String id) {
        try {
            List<Event> events = new ArrayList<>();
            events.add(Event.S3_OBJECT_CREATED_PUT);

            TopicConfiguration config = TopicConfiguration.builder()
                    .topicArn(topicArn)
                    .events(events)
                    .id(id)
                    .build();

            List<TopicConfiguration> topics = new ArrayList<>();
```

```
            topics.add(config);

            NotificationConfiguration configuration =
 NotificationConfiguration.builder()
                    .topicConfigurations(topics)
                    .build();

            PutBucketNotificationConfigurationRequest configurationRequest =
 PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
                    .build();

            // Set the bucket notification configuration.
            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
 events enabled.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see PutBucketNotificationConfiguration in *AWS SDK for Java 2.x API Reference*.

PutBucketPolicy

The following code example shows how to use `PutBucketPolicy`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <polFile>

                Where:
                    bucketName - The Amazon S3 bucket to set the policy on.
                    polFile - A JSON file containing the policy (see the Amazon S3
 Readme for an example).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```
                .region(region)
                .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }

    public static void setPolicy(S3Client s3, String bucketName, String policyText)
{
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();

            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }

    // Loads a JSON-formatted policy from a file
    public static String getBucketPolicyFromFile(String policyFile) {

        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"", policyFile);
```

```
            System.out.println(e.getMessage());
        }

        try {
            final JsonParser parser = new
 ObjectMapper().getFactory().createParser(fileText.toString());
            while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
        }
        return fileText.toString();
    }
}
```

- For API details, see PutBucketPolicy in *AWS SDK for Java 2.x API Reference*.

PutBucketWebsite

The following code example shows how to use `PutBucketWebsite`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucketName> [indexdoc]\s

                Where:
                   bucketName   - The Amazon S3 bucket to set the website
 configuration on.\s
                   indexdoc - The index document, ex. 'index.html'
                              If not specified, 'index.html' will be set.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    public static void setWebsiteConfig(S3Client s3, String bucketName, String
 indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
                    .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                    .build();

            PutBucketWebsiteRequest pubWebsiteReq =
 PutBucketWebsiteRequest.builder()
                    .bucket(bucketName)
```

```
                    .websiteConfiguration(websiteConfig)
                    .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see PutBucketWebsite in *AWS SDK for Java 2.x API Reference*.

`PutObject`

The following code example shows how to use `PutObject`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Upload a file to a bucket using an S3Client.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <objectKey> <objectPath>\s

                Where:
                  bucketName - The Amazon S3 bucket to upload an object into.
                  objectKey - The object to upload (for example, book.pdf).
                  objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
 String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
```

```
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
 + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Use an S3TransferManager to upload a file to a bucket. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
```

```
        return uploadResult.response().eTag();
    }
```

Upload an object to a bucket and set tags using an [S3Client](https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/s3/S3Client.html).

```
    public static void putS3ObjectTags(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Tag tag1 = Tag.builder()
                    .key("Tag 1")
                    .value("This is tag 1")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Tag 2")
                    .value("This is tag 2")
                    .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag1);
            tags.add(tag2);

            Tagging allTags = Tagging.builder()
                    .tagSet(tags)
                    .build();

            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .tagging(allTags)
                    .build();

            s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateObjectTags(S3Client s3, String bucketName, String
 objectKey) {
```

```
        try {
            GetObjectTaggingRequest taggingRequest =
 GetObjectTaggingRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            GetObjectTaggingResponse getTaggingRes =
 s3.getObjectTagging(taggingRequest);
            List<Tag> obTags = getTaggingRes.tagSet();
            for (Tag sinTag : obTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

            // Replace the object's tags with two new tags.
            Tag tag3 = Tag.builder()
                    .key("Tag 3")
                    .value("This is tag 3")
                    .build();

            Tag tag4 = Tag.builder()
                    .key("Tag 4")
                    .value("This is tag 4")
                    .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag3);
            tags.add(tag4);

            Tagging updatedTags = Tagging.builder()
                    .tagSet(tags)
                    .build();

            PutObjectTaggingRequest taggingRequest1 =
 PutObjectTaggingRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .tagging(updatedTags)
                    .build();

            s3.putObjectTagging(taggingRequest1);
            GetObjectTaggingResponse getTaggingRes2 =
 s3.getObjectTagging(taggingRequest);
```

```
            List<Tag> modTags = getTaggingRes2.tagSet();
            for (Tag sinTag : modTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Return a byte array.
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        return bytesArray;
    }
}
```

Upload an object to a bucket and set metadata using an S3Client.

```
import software.amazon.awssdk.core.sync.RequestBody;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = """

                Usage:
                  <bucketName> <objectKey> <objectPath>\s

                Where:
                  bucketName - The Amazon S3 bucket to upload an object into.
                  objectKey - The object to upload (for example, book.pdf).
                  objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
 bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
```

```
                .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
 String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
 + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Upload an object to a bucket and set an object retention value using an [S3Client](https://docs.aws.amazon.com/goto/SdkForJavaV2/s3-2006-03-01/S3Client).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
```

```
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <key> <bucketName>\s

                Where:
                    key - The name of the object (for example, book.pdf).\s
                    bucketName - The Amazon S3 bucket name that contains the object
 (for example, bucket1).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    public static void setRentionPeriod(S3Client s3, String key, String bucket) {
        try {
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
```

```
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                    .mode("COMPLIANCE")
                    .retainUntilDate(instant)
                    .build();

            PutObjectRetentionRequest retentionRequest =
 PutObjectRetentionRequest.builder()
                    .bucket(bucket)
                    .key(key)
                    .bypassGovernanceRetention(true)
                    .retention(lockRetention)
                    .build();

            // To set Retention on an object, the Amazon S3 bucket must support
 object
            // locking, otherwise an exception is thrown.
            s3.putObjectRetention(retentionRequest);
            System.out.print("An object retention configuration was successfully
 placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see PutObject in *AWS SDK for Java 2.x API Reference*.

PutObjectLegalHold

The following code example shows how to use `PutObjectLegalHold`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
 legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
 PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
 +".");
    }
```

- For API details, see PutObjectLegalHold in *AWS SDK for Java 2.x API Reference*.

PutObjectLockConfiguration

The following code example shows how to use `PutObjectLockConfiguration`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Set the object lock configuration of a bucket.

```
    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
 VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
 PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
 PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on "+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage() +
 "'");
        }
    }
```

Set the default retention period of a bucket.

```
    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
 VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
```

```
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest versioningRequest =
 PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            getClient().putBucketVersioning(versioningRequest);
            DefaultRetention rention = DefaultRetention.builder()
                .days(1)
                .mode(ObjectLockRetentionMode.GOVERNANCE)
                .build();

            ObjectLockRule lockRule = ObjectLockRule.builder()
                .defaultRetention(rention)
                .build();

            ObjectLockConfiguration objectLockConfiguration =
 ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
                .rule(lockRule)
                .build();

            PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
 PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(objectLockConfiguration)
                .build();

            getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
            System.out.println("Added a default retention to bucket "+bucketName +".");
        }
```

- For API details, see PutObjectLockConfiguration in *AWS SDK for Java 2.x API Reference*.

PutObjectRetention

The following code example shows how to use `PutObjectRetention`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/s3#code-examples).

```
    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time zone.
        ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
 HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
 PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
 until "+ humanReadableDate +".");
    }
```

- For API details, see PutObjectRetention in *AWS SDK for Java 2.x API Reference*.

RestoreObject

The following code example shows how to use `RestoreObject`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 *  For more information about restoring an object, see "Restoring an archived
 object" at
 *  https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 *  Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <expectedBucketOwner>

                Where:
```

```
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name of an object with a Storage class value
 of Glacier.\s
                    expectedBucketOwner - The account that owns the bucket (you can
 obtain this value from the AWS Management Console).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    public static void restoreS3Object(S3Client s3, String bucketName, String
 keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                    .days(10)

 .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                    .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                    .expectedBucketOwner(expectedBucketOwner)
                    .bucket(bucketName)
                    .key(keyName)
                    .restoreRequest(restoreRequest)
                    .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
                System.exit(1);
            }
        }
    }
```

- For API details, see [RestoreObject](#) in *AWS SDK for Java 2.x API Reference*.

`SelectObjectContent`

The following code example shows how to use `SelectObjectContent`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

The following example shows a query using a JSON object. The [complete example](#) also shows the use of a CSV object.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
```

```
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
 LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" + UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
 SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where c.area
 < 350000";
        final String fileType = FILE_JSON;
```

```
        InputSerialization inputSerialization = InputSerialization.builder()
                .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
                .compressionType(CompressionType.NONE)
                .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
                .json(JSONOutput.builder().recordDelimiter(null).build())
                .build();

        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
                .bucket(BUCKET_NAME)
                .key(FILE_JSON)
                .expression(queryExpression)
                .expressionType(ExpressionType.SQL)
                .inputSerialization(inputSerialization)
                .outputSerialization(outputSerialization)
                .build();

        EventStreamInfo eventStreamInfo = new EventStreamInfo();
        // Call the selectObjectContent method with the request and a response
 handler.
        // Supply an EventStreamInfo object to the response handler to gather
 records and information from the response.
        s3AsyncClient.selectObjectContent(select,
 buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record ->
                record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times", fileType,
 eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
 eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
 eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler buildResponseHandler(EventStreamInfo
 eventStreamInfo) {
```

```
        // Use a Visitor to process the response stream. This visitor logs
 information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
 SelectObjectContentResponseHandler.Visitor.builder()
                .onRecords(r -> {
                    logger.info("Record event received.");
                    eventStreamInfo.addRecord(r.payload().asUtf8String());
                    eventStreamInfo.incrementOnRecordsCalled();
                })
                .onCont(ce -> {
                    logger.info("Continuation event received.");
                    eventStreamInfo.incrementContinuationEvents();
                })
                .onProgress(pe -> {
                    Progress progress = pe.details();
                    logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                            progress.bytesScanned(),
                            progress.bytesProcessed(),
                            progress.bytesReturned());
                })
                .onEnd(ee -> logger.info("End event received."))
                .onStats(se -> {
                    logger.info("Stats event received.");
                    eventStreamInfo.addStats(se.details());
                })
                .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
 processes the stream.
        return SelectObjectContentResponseHandler.builder()
                .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
 processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
```

```
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
            records.add(record);
        }

        void addStats(Stats stats) {
            this.stats = stats;
        }

        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }

        public Stats getStats() {
            return stats;
        }
    }
```

- For API details, see SelectObjectContent in *AWS SDK for Java 2.x API Reference*.

Scenarios

Create a presigned URL

The following code example shows how to create a presigned URL for Amazon S3 and upload an object.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/s3#code-examples).

Generate a pre-signed URL for an object, then download it (GET request).

Imports.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

Generate the URL.

```
    /* Create a pre-signed URL to download an object in a subsequent GET request. */
    public String createPresignedGetUrl(String bucketName, String keyName) {
        try (S3Presigner presigner = S3Presigner.create()) {

            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            GetObjectPresignRequest presignRequest =
 GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))  // The URL will
 expire in 10 minutes.
                    .getObjectRequest(objectRequest)
                    .build();

            PresignedGetObjectRequest presignedRequest =
 presigner.presignGetObject(presignRequest);
            logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
            logger.info("HTTP method: [{}]",
 presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
```

Download the object by using any one of the following three approaches.

Use JDK `HttpURLConnection` (since v1.1) class to do the download.

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
    public byte[] useHttpUrlConnectionToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
 ByteArrayOutputStream(); // Capture the response body to a byte array.

        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
 presignedUrl.openConnection();
            connection.setRequestMethod("GET");
```

```
            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                IoUtils.copy(content, byteArrayOutputStream);
            }
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Use JDK `HttpClient` (since v11) class to do the download.

```
    /* Use the JDK HttpClient (since v11) class to do the download. */
    public byte[] useHttpClientToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
 ByteArrayOutputStream(); // Capture the response body to a byte array.

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpResponse<InputStream> response = httpClient.send(requestBuilder
                            .uri(presignedUrl.toURI())
                            .GET()
                            .build(),
                    HttpResponse.BodyHandlers.ofInputStream());

            IoUtils.copy(response.body(), byteArrayOutputStream);

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Use the AWS SDK for Java `SdkHttpClient` class to do the download.

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToPut(String presignedUrlString) {

        ByteArrayOutputStream byteArrayOutputStream = new
 ByteArrayOutputStream(); // Capture the response body to a byte array.
        try {
            URL presignedUrl = new URL(presignedUrlString);
            SdkHttpRequest request = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.GET)
                    .uri(presignedUrl.toURI())
                    .build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
 sdkHttpClient.prepareRequest(executeRequest).call();
                response.responseBody().ifPresentOrElse(
                        abortableInputStream -> {
                            try {
                                IoUtils.copy(abortableInputStream,
 byteArrayOutputStream);
                            } catch (IOException e) {
                                throw new RuntimeException(e);
                            }
                        },
                        () -> logger.error("No response body."));

                logger.info("HTTP Response code is {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Generate a pre-signed URL for an upload, then upload a file (PUT request).

Imports.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Generate the URL.

```
    /* Create a presigned URL to use in a subsequent PUT request */
    public String createPresignedUrl(String bucketName, String keyName, Map<String,
 String> metadata) {
        try (S3Presigner presigner = S3Presigner.create()) {

            PutObjectRequest objectRequest = PutObjectRequest.builder()
```

```
                    .bucket(bucketName)
                    .key(keyName)
                    .metadata(metadata)
                    .build();

            PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))  // The URL expires
in 10 minutes.
                    .putObjectRequest(objectRequest)
                    .build();


            PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
            String myURL = presignedRequest.url().toString();
            logger.info("Presigned URL to upload a file to: [{}]", myURL);
            logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
```

Upload a file object by using any one of the following three approaches.

Use the JDK `HttpURLConnection` (since v1.1) class to do the upload.

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
    public void useHttpUrlConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" +
k, v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
```

```
             FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
                buffer.clear();
            }
        } catch (IOException e) {
            logger.error(e.getMessage(), e);
        }

        out.close();
        connection.getResponseCode();
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

Use the JDK `HttpClient` (since v11) class to do the upload.

```
    /* Use the JDK HttpClient (since v11) class to do the upload. */
    public void useHttpClientToPut(String presignedUrlString, File fileToPut,
 Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            final HttpResponse<Void> response = httpClient.send(requestBuilder
                            .uri(new URL(presignedUrlString).toURI())

 .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
                            .build(),
                    HttpResponse.BodyHandlers.discarding());
```

```
            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Use the AWS for Java V2 `SdkHttpClient` class to do the upload.

```
    /* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
    public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
 Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        try {
            URL presignedUrl = new URL(presignedUrlString);

            SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.PUT)
                    .uri(presignedUrl.toURI());
            // Add headers
            metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k,
 v));
            // Finish building the request.
            SdkHttpRequest request = requestBuilder.build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
 FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
 sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Delete incomplete multipart uploads

The following code example shows how to how to delete or stop incomplete Amazon S3 multipart uploads.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/s3#code-examples).

To stop multipart uploads that are in-progress or incomplete for any reason, you can get a list uploads and then delete them as shown in the following example.

```
    public static void abortIncompleteMultipartUploadsFromList() {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
```

```
                logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
upload.uploadId(), bucketName);
            }
        }
    }
```

To delete incomplete multipart uploads that were initiated before or after a date, you can selectively delete multipart uploads based on a point in time as shown in the following example.

```
    static void abortIncompleteMultipartUploadsOlderThan(Instant pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated [{}]",
upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(upload.key())
                        .expectedBucketOwner(accountId)
                        .uploadId(upload.uploadId())
                        .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }
```

If you have access to the upload ID after you begin a multipart upload, you can delete the in-progress upload by using the ID.

```
    static void abortMultipartUploadUsingUploadId() {
        String uploadId = startUploadReturningUploadId();
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -> b
                .uploadId(uploadId)
                .bucket(bucketName)
                .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
        }
    }
```

To consistently delete incomplete multipart uploads older that a certain number of days, set up a bucket lifecycle configuration for the bucket. The following example shows how to create a rule to delete incomplete uploads older than 7 days.

```
    static void abortMultipartUploadsUsingLifecycleConfig() {
        Collection<LifecycleRule> lifeCycleRules = List.of(LifecycleRule.builder()
                .abortIncompleteMultipartUpload(b -> b.
                        daysAfterInitiation(7))
                .status("Enabled")
                .filter(SdkBuilder::build) // Filter element is required.
                .build());

        // If the action is successful, the service sends back an HTTP 200 response
with an empty HTTP body.
        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
                .bucket(bucketName)
                .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
        } else {
```

```
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AbortMultipartUpload
  - ListMultipartUploads
  - PutBucketLifecycleConfiguration

Download objects to a local directory

The following code example shows how to download all objects in an Amazon Simple Storage Service (Amazon S3) bucket to a local directory.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Use an S3TransferManager to download all S3 objects in the same S3 bucket. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
```

```
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
 fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- For API details, see DownloadDirectory in *AWS SDK for Java 2.x API Reference*.

Get started with buckets and objects

The following code example shows how to:

- Create a bucket and upload a file to it.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the bucket objects and the bucket.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Deletes the object from the Amazon S3 bucket.
 * 8. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <bucketName> <key> <objectPath> <savePath> <toBucket>

                Where:
                    bucketName - The Amazon S3 bucket to create.
                    key - The key to use.
                    objectPath - The path where the file is located (for example,
 C:/AWS/book2.pdf).
```

```
                    savePath - The path where the file is saved after it's
downloaded (for example, C:/AWS/book2.pdf).
                    toBucket - An Amazon S3 bucket to where an object is copied to
(for example, C:/AWS/book2.pdf).\s
                    """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an Amazon S3 bucket.");
        createBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update a local file to the Amazon S3 bucket.");
        uploadLocalFile(s3, bucketName, key, objectPath);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Download the object to another local file.");
        getObjectBytes(s3, bucketName, key, savePath);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Perform a multipart upload.");
        String multipartKey = "multiPartKey";
        multipartUpload(s3, toBucket, multipartKey);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List all objects located in the Amazon S3 bucket.");
        listAllObjects(s3, bucketName);
        anotherListExample(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Copy the object to another Amazon S3 bucket.");
        copyBucketObject(s3, bucketName, key, toBucket);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the object from the Amazon S3 bucket.");
        deleteObjectFromBucket(s3, bucketName, key);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Delete the Amazon S3 bucket.");
        deleteBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon S3 operations were successfully performed");
        System.out.println(DASHES);
        s3.close();
    }

    // Create a bucket by using a S3Waiter object.
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
```

```
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteBucket(S3Client client, String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucket)
                .build();

        client.deleteBucket(deleteBucketRequest);
        System.out.println(bucket + " was deleted.");
    }

    /**
     * Upload an object in parts.
     */
    public static void multipartUpload(S3Client s3, String bucketName, String key) {
        int mB = 1024 * 1024;
        // First create a multipart upload and get the upload id.
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object.
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(1).build();
```

```
        String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
                .eTag();
        CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
                .uploadId(uploadId)
                .partNumber(2).build();
        String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
                .eTag();
        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Call completeMultipartUpload operation to tell S3 to merge all uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
                .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .multipartUpload(completedMultipartUpload)
                .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        byte[] b = new byte[size];
        new Random().nextBytes(b);
        return ByteBuffer.wrap(b);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
```

```
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
    }

    public static void listAllObjects(S3Client s3, String bucketName) {
        ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

        boolean done = false;
        while (!done) {
            ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
```

```
            for (S3Object content : listObjResponse.contents()) {
                System.out.println(content.key());
            }

            if (listObjResponse.nextContinuationToken() == null) {
                done = true;
            }

            listObjectsReqManual = listObjectsReqManual.toBuilder()
                    .continuationToken(listObjResponse.nextContinuationToken())
                    .build();
        }
    }

    public static void anotherListExample(S3Client s3, String bucketName) {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

        // Process response pages.
        listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

        // Helper method to work with paginated collection of items directly.
        listRes.contents().stream()
                .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

        for (S3Object content : listRes.contents()) {
            System.out.println(" Key: " + content.key() + " size = " +
content.size());
        }
    }

    public static void deleteObjectFromBucket(S3Client s3, String bucketName, String
key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
```

```
                .build();

        s3.deleteObject(deleteObjectRequest);
        System.out.println(key + " was deleted");
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
 objectKey, String toBucket) {
        String encodedUrl = null;
        try {
            encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
 StandardCharsets.UTF_8.toString());
        } catch (UnsupportedEncodingException e) {
            System.out.println("URL could not be encoded: " + e.getMessage());
        }
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
                .copySource(encodedUrl)
                .destinationBucket(toBucket)
                .destinationKey(objectKey)
                .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            System.out.println("The " + objectKey + " was copied to " + toBucket);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject

- ListObjectsV2
- PutObject

Get the legal hold configuration of an object

The following code example shows how to get the legal hold configuration of an S3 bucket.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
 objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
 GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
 getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
 bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
 "'");
        }

        return null;
    }
```

- For API details, see GetObjectLegalHold in *AWS SDK for Java 2.x API Reference*.

Lock Amazon S3 objects

The following code example shows how to work with S3 object lock features.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run an interactive scenario demonstrating Amazon S3 object lock features.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
 Before running this Java V2 code example, set up your development
 environment, including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

 This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
 policies.
    2. Upload sample objects to each bucket.
    3. Set some Legal Hold and Retention Periods on objects and buckets.
    4. Investigate lock policies by viewing settings or attempting to delete or
 overwrite objects.
    5. Clean up objects and buckets.
```

```
 */
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        // Get the current date and time to ensure bucket name is unique.
        LocalDateTime currentTime = LocalDateTime.now();

        // Format the date and time as a string.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
        String timeStamp = currentTime.format(formatter);

        s3LockActions = new S3LockActions();
        bucketName = "bucket"+timeStamp;
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3) Object
 Locking Workflow Scenario.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
```

```
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by selecting
 one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to delete:");
                    List<S3InfoObject> allFiles =
 s3LockActions.listBucketsAndObjects(bucketNames, true);
```

```
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
                }

                case 2 -> {
                    System.out.println("Enter the number of the object to delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
                }

                case 3 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();

                    // Attempt to overwrite the file.
                    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
                        writer.write("This is a modified text.");

                    } catch (IOException e) {
```

```
                        e.printStackTrace();
                    }
                    s3LockActions.uploadFile(bucketName, objectKey, objectKey);
                }

                case 4 -> {
                    System.out.println("Enter the number of the object to
 overwrite:");
                    List<S3InfoObject> allFiles =
 s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
 f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectRetention(bucketName, objectKey);
                }

                case 5 -> {
                    System.out.println("Enter the number of the object to view:");
                    List<S3InfoObject> allFiles =
 s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
 f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectLegalHold(bucketName, objectKey);
                    s3LockActions.getBucketObjectLockConfiguration(bucketName);
                }

                case 6 -> {
                    System.out.println("Exiting the workflow...");
                    return;
                }

                default -> {
                    System.out.println("Invalid choice. Please select again.");
                }
            }
        }
    }
```

```
    // Clean up the resources from the scenario.
    private static void cleanup() {
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
        for (S3InfoObject fileInfo : allFiles) {
            String bucketName = fileInfo.getBucketName();
            String key = fileInfo.getKeyName();
            String version = fileInfo.getVersion();
            if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
                ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
                if (legalHold != null) {
                    String holdStatus = legalHold.status().name();
                    System.out.println(holdStatus);
                    if (holdStatus.compareTo("ON") == 0) {
                        s3LockActions.modifyObjectLegalHold(bucketName, key, false);
                    }
                }
                // Check for a retention period.
                ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
                boolean hasRetentionPeriod ;
                hasRetentionPeriod = retention != null;
                s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

            } else {
                System.out.println(bucketName +" objects do not have a legal lock");
                s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
            }
        }

        // Delete the buckets.
        System.out.println("Delete "+bucketName);
        for (String bucket : bucketNames){
            s3LockActions.deleteBucketByName(bucket);
        }
    }

    private static void setup() {
        Scanner scanner = new Scanner(System.in);
```

```
        System.out.println("""
                For this workflow, we will use the AWS SDK for Java to create
 several S3
                buckets and files to demonstrate working with S3 locking features.
                """);

        System.out.println("S3 buckets can be created either with or without object
 lock enabled.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();

        // Create three S3 buckets.
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
        s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to use
 object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Object lock policies can also be added to existing
 buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
 java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
```

```
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
                    fileNameWithoutExtension = fileNameWithoutExtension.substring(0,
extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on individual
files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
                                System.out.println("**** You have selected to put a
legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                            }
                        }
                        case 1 -> {
```

```
                                """
                                    Would you like to add a 1 day Governance retention
 period to %s in %s (y/n)?
                                    Reminder: Only a user with the
 s3:BypassGovernanceRetention permission will be able to delete this file or its
 bucket until the retention period has expired.
                                    """.formatted(exampleFileName, bucketName);
                                System.out.println(question);
                                String ans2 = scanner.nextLine().trim();
                                if (ans2.equalsIgnoreCase("y")) {

 s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                                }
                            }
                        }
                    }
                }
            }
        }

    // Get file extension.
    private static String getFileExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0) {
            return fileName.substring(dotIndex);
        }
        return "";
    }

    public static void configurationSetup() {
        String noLockBucketName = bucketName + "-no-lock";
        String lockEnabledBucketName = bucketName + "-lock-enabled";
        String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
```

```
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }

        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

A wrapper class for S3 functions.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
```

```
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time zone.
        ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
 HH:mm:ss");
```

```
        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }

    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
"'");
        }
```

```
        return null;
    }

    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
                }))
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the updated
value.
                if (interactive) {
                    System.out.println(i + ": "+ s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
```

```
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
 bucketName) {
        ListObjectVersionsRequest versionsRequest =
 ListObjectVersionsRequest.builder()
            .bucket(bucketName)
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
 VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
 PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
 ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();
```

```
        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();

        getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName +".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on "+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage() +
"'");
        }
    }
```

```
    public void uploadFile(String bucketName, String objectName, String filePath) {
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectName)
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)
            .build();

        PutObjectResponse response = getClient().putObject(request, file);
        if (response != null) {
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +
 bucketName + ".");
        } else {
            System.out.println("\tCould not upload " + objectName + " to " +
 bucketName + ".");
        }
    }

    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
 legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
 PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
 +".");
    }
```

```
    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey, boolean
 hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
 GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
 getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +" in "+ bucketName +":
 " + response.retention().mode() +" until "+ response.retention().retainUntilDate()
 +".");
            return response.retention();
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }

    public void deleteBucketByName(String bucketName) {
        try {
            DeleteBucketRequest request = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

            getClient().deleteBucket(request);
            System.out.println(bucketName +" was deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
 GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
 getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
 "+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
 response.objectLockConfiguration().rule().defaultRetention());
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention

- PutObjectLegalHold
- PutObjectLockConfiguration
- PutObjectRetention

Parse URIs

The following code example shows how to parse Amazon S3 URIs to extract important components like the bucket name and object key.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Parse an Amazon S3 URI by using the S3Uri class.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

    /**
     *
     * @param s3Client    - An S3Client through which you acquire an S3Uri instance.
     * @param s3ObjectUrl - A complex URL (String) that is used to demonstrate S3Uri
     *                      capabilities.
     */
    public static void parseS3UriExample(S3Client s3Client, String s3ObjectUrl) {
        logger.info(s3ObjectUrl);
        // Console output:
        // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.
```

```
        // Create an S3Utilities object using the configuration of the s3Client.
        S3Utilities s3Utilities = s3Client.utilities();

        // From a String URL create a URI object to pass to the parseUri() method.
        URI uri = URI.create(s3ObjectUrl);
        S3Uri s3Uri = s3Utilities.parseUri(uri);

        // If the URI contains no value for the Region, bucket or key, the SDK
returns
        // an empty Optional.
        // The SDK returns decoded URI values.

        Region region = s3Uri.region().orElse(null);
        log("region", region);
        // Console output: 'region: us-west-1'.

        String bucket = s3Uri.bucket().orElse(null);
        log("bucket", bucket);
        // Console output: 'bucket: myBucket'.

        String key = s3Uri.key().orElse(null);
        log("key", key);
        // Console output: 'key: resources/doc.txt'.

        Boolean isPathStyle = s3Uri.isPathStyle();
        log("isPathStyle", isPathStyle);
        // Console output: 'isPathStyle: true'.

        // If the URI contains no query parameters, the SDK returns an empty map.
        Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
        log("rawQueryParameters", queryParams);
        // Console output: 'rawQueryParameters: {versionId=[abc123], partNumber=[77,
        // 88]}'.

        // Retrieve the first or all values for a query parameter as shown in the
        // following code.
        String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
        log("firstMatchingRawQueryParameter-versionId", versionId);
        // Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

        String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
```

```
        log("firstMatchingRawQueryParameter-partNumber", partNumber);
        // Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

        List<String> partNumbers =
 s3Uri.firstMatchingRawQueryParameters("partNumber");
        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters, must
 be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".", the
 dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
```

Perform a multipart upload

The following code example shows how to perform a multipart upload to an Amazon S3 object.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples).

The code examples use the following imports.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

Use the S3 Transfer Manager on top of the AWS CRT-based S3 client to transparently perform a multipart upload when the size of the content exceeds a threshold. The default threshold size is 8 MB.

```
    public void multipartUploadWithTransferManager(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b
                        .bucket(bucketName)
                        .key(key))
                .source(Paths.get(filePath))
                .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

Use the S3Client API to perform a multipart upload.

```
    public void multipartUploadWithS3Client(String filePath) {

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
 s3Client.createMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key));
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
 buffer.
```

```
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            logger.error(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)

 .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

Use the S3AsyncClient API with multipart support enabled to perform a multipart upload.

```
    public void multipartUploadWithS3AsyncClient(String filePath) {
        // Enable multipart support.
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
                .multipartEnabled(true)
                .build();
```

```
        CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b ->
b
                        .bucket(bucketName)
                        .key(key),
                Paths.get(filePath));

        response.join();
        logger.info("File uploaded in multiple 8 MiB parts using S3AsyncClient.");
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

Track uploads and downloads

The following code example shows how to track an Amazon S3 object upload or download.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Track the progress of a file upload.

```
    public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                                String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())  // Add
listener.
                .source(Paths.get(filePathURI))
                .build();
```

```
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        fileUpload.completionFuture().join();
        /*
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
                        <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3 MB
file upload.
                Transfer initiated...
                |                    | 0.0%
                |====                | 21.1%
                |============        | 60.5%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

Track the progress of a file download.

```
    public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                                  String key, String downloadedFileWithPath) {
```

```
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())  // Add
 listener.
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
 transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
 downloadFile.completionFuture().join();
        /*
            The SDK provides a LoggingTransferListener implementation of the
 TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
                        <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
 name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
 level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3 MB
 file download.
                Transfer initiated...
                |=======             | 39.4%
                |===============     | 78.8%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - GetObject
  - PutObject

Upload directory to a bucket

The following code example shows how to upload a local directory recursively to an Amazon Simple Storage Service (Amazon S3) bucket.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Use an S3TransferManager to upload a local directory. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public Integer uploadDirectory(S3TransferManager transferManager,
            URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
 transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
```

```
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- For API details, see UploadDirectory in *AWS SDK for Java 2.x API Reference*.

Upload or download large files

The following code example shows how to upload or download large files to and from Amazon S3.

For more information, see Uploading an object using multipart upload.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Call functions that transfer files to and from an S3 bucket using the S3TransferManager.

```
    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
```

```
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

Upload an entire local directory.

```
    public Integer uploadDirectory(S3TransferManager transferManager,
            URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

Upload a single file.

```
    public String uploadFile(S3TransferManager transferManager, String bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Upload stream of unknown size

The following code example shows how to upload a stream of unknown size to an Amazon S3 object.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Use the AWS CRT-based S3 Client.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

    /**
     * @param s33CrtAsyncClient - To upload content from a stream of unknown size,
 use the AWS CRT-based S3 client. For more information, see
     *                          https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/crt-based-s3-client.html.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
 metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStream(S3AsyncClient s33CrtAsyncClient,
 String bucketName, String key) {

        BlockingInputStreamAsyncRequestBody body =
```

```
                AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
 stream will be provided later.

        CompletableFuture<PutObjectResponse> responseFuture =
                s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
 body);

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        return response;
    }
}
```

Use the Amazon S3 Transfer Manager.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.util.UUID;

    /**
     * @param transferManager - To upload content from a stream of unknown size, use
 the S3TransferManager based on the AWS CRT-based S3 client.
```

```
     *                        For more information, see https://docs.aws.amazon.com/
sdk-for-java/latest/developer-guide/transfer-manager.html.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
 bucketName, String key) {

        BlockingInputStreamAsyncRequestBody body =
                AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
 stream will be provided later.

        Upload upload = transferManager.upload(builder -> builder
                .requestBody(body)
                .putObjectRequest(req -> req.bucket(bucketName).key(key))
                .build());

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        return upload.completionFuture().join();
    }
}
```

Use checksums

The following code example shows how to use checksums to work with an Amazon S3 object.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

The code examples use a subset of the following imports.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
```

```
import java.util.UUID;
```

Specify a checksum algorithm for the `putObject` method when you [build the `PutObjectRequest`](#).

```
    public void putObjectWithChecksum() {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.CRC32),
                RequestBody.fromString("This is a test"));
    }
```

Verify the checksum for the `getObject` method when you [build the GetObjectRequest](#).

```
    public GetObjectResponse getObjectWithChecksum() {
        return s3Client.getObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumMode(ChecksumMode.ENABLED))
                .response();
    }
```

Pre-calculate a checksum for the `putObject` method when you [build the `PutObjectRequest`](#).

```
    public void putObjectWithPrecalculatedChecksum(String filePath) {
        String checksum = calculateChecksum(filePath, "SHA-256");

        s3Client.putObject((b -> b
                .bucket(bucketName)
                .key(key)
                .checksumSHA256(checksum)),
                RequestBody.fromFile(Paths.get(filePath)));
    }
```

Use the S3 Transfer Manager on top of the AWS CRT-based S3 client to transparently perform a multipart upload when the size of the content exceeds a threshold. The default threshold size is 8 MB.

You can specify a checksum algorithm for the SDK to use. By default, the SDK uses the CRC32 algorithm.

```
    public void multipartUploadWithChecksumTm(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b
                        .bucket(bucketName)
                        .key(key)
                        .checksumAlgorithm(ChecksumAlgorithm.SHA1))
                .source(Paths.get(filePath))
                .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

Use the S3Client API or (S3AsyncClient API) to perform a multipart upload. If you specify an additional checksum, you must specify the algorithm to use on the initiation of the upload. You must also specify the algorithm for each part request and provide the checksum calculated for each part after it is uploaded.

```
    public void multipartUploadWithChecksumS3Client(String filePath) {
        ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
  s3Client.createMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer
```

```
        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .checksumAlgorithm(algorithm) // Checksum specified on each
part.
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
```

```
.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

Serverless examples

Invoke a Lambda function from an Amazon S3 trigger

The following code example shows how to implement a Lambda function that receives an event triggered by uploading an object to an S3 bucket. The function retrieves the S3 bucket name and object key from the event parameter and calls the Amazon S3 API to retrieve and log the content type of the object.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an S3 event with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
```

```
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

          logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

          return "Ok";
        } catch (Exception e) {
          throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

S3 Glacier examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with S3 Glacier.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

CreateVault

The following code example shows how to use `CreateVault`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class CreateVault {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to create.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            CreateVaultResponse createVaultResult =
 glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
 createVaultResult.location());

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see CreateVault in *AWS SDK for Java 2.x API Reference*.

DeleteArchive

The following code example shows how to use `DeleteArchive`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteArchive {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName> <accountId> <archiveId>

                Where:
                   vaultName - The name of the vault that contains the archive to
 delete.
                   accountId - The account ID value.
                   archiveId - The archive ID value.
                """;

        if (args.length != 3) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String archiveId = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
        glacier.close();
    }

    public static void deleteGlacierArchive(GlacierClient glacier, String vaultName,
 String accountId,
            String archiveId) {
        try {
            DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
                    .vaultName(vaultName)
                    .accountId(accountId)
                    .archiveId(archiveId)
                    .build();

            glacier.deleteArchive(delArcRequest);
            System.out.println("The archive was deleted.");

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteArchive in *AWS SDK for Java 2.x API Reference*.

DeleteVault

The following code example shows how to use `DeleteVault`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteVault {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        deleteGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void deleteGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            glacier.deleteVault(delVaultRequest);
            System.out.println("The vault was deleted!");

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteVault in *AWS SDK for Java 2.x API Reference*.

InitiateJob

The following code example shows how to use `InitiateJob`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Retrieve a vault inventory.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
```

```
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ArchiveDownload {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <vaultName> <accountId> <path>

                Where:
                   vaultName - The name of the vault.
                   accountId - The account ID value.
                   path - The path where the file is written to.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String path = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        String jobNum = createJob(glacier, vaultName, accountId);
        checkJob(glacier, jobNum, vaultName, accountId, path);
        glacier.close();
    }

    public static String createJob(GlacierClient glacier, String vaultName, String
 accountId) {
        try {
            JobParameters job = JobParameters.builder()
                    .type("inventory-retrieval")
                    .build();

            InitiateJobRequest initJob = InitiateJobRequest.builder()
                    .jobParameters(job)
                    .accountId(accountId)
                    .vaultName(vaultName)
                    .build();

            InitiateJobResponse response = glacier.initiateJob(initJob);
            System.out.println("The job ID is: " + response.jobId());
            System.out.println("The relative URI path of the job is: " +
 response.location());
            return response.jobId();

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);

        }
        return "";
    }

    // Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
    // Documentation.
    public static void checkJob(GlacierClient glacier, String jobId, String name,
 String account, String path) {
        try {
            boolean finished = false;
            String jobStatus;
            int yy = 0;

            while (!finished) {
                DescribeJobRequest jobRequest = DescribeJobRequest.builder()
                        .jobId(jobId)
```

```
                        .accountId(account)
                        .vaultName(name)
                        .build();

                DescribeJobResponse response = glacier.describeJob(jobRequest);
                jobStatus = response.statusCodeAsString();

                if (jobStatus.compareTo("Succeeded") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + jobStatus);
                    Thread.sleep(1000);
                }
                yy++;
            }

            System.out.println("Job has Succeeded");
            GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
                    .jobId(jobId)
                    .vaultName(name)
                    .accountId(account)
                    .build();

            ResponseBytes<GetJobOutputResponse> objectBytes =
 glacier.getJobOutputAsBytes(jobOutputRequest);
            // Write the data to a local file.
            byte[] data = objectBytes.asByteArray();
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from a Glacier vault");
            os.close();

        } catch (GlacierException | InterruptedException | IOException e) {
            System.out.println(e.getMessage());
            System.exit(1);

        }
    }
}
```

- For API details, see InitiateJob in *AWS SDK for Java 2.x API Reference*.

ListVaults

The following code example shows how to use `ListVaults`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/glacier#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListVaults {
    public static void main(String[] args) {
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllVault(glacier);
        glacier.close();
    }

    public static void listAllVault(GlacierClient glacier) {
        boolean listComplete = false;
        String newMarker = null;
        int totalVaults = 0;
        System.out.println("Your Amazon Glacier vaults:");
```

```
        try {
            while (!listComplete) {
                ListVaultsResponse response = null;
                if (newMarker != null) {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = glacier.listVaults(request);
                } else {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .build();
                    response = glacier.listVaults(request);
                }

                List<DescribeVaultOutput> vaultList = response.vaultList();
                for (DescribeVaultOutput v : vaultList) {
                    totalVaults += 1;
                    System.out.println("* " + v.vaultName());
                }

                // Check for further results.
                newMarker = response.marker();
                if (newMarker == null) {
                    listComplete = true;
                }
            }

            if (totalVaults == 0) {
                System.out.println("No vaults found.");
            }

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListVaults in *AWS SDK for Java 2.x API Reference*.

UploadArchive

The following code example shows how to use UploadArchive.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UploadArchive {

    static final int ONE_MB = 1024 * 1024;

    public static void main(String[] args) {
        final String usage = """

                Usage:   <strPath> <vaultName>\s

                Where:
```

```
                    strPath - The path to the archive to upload (for example, C:\\AWS
\\test.pdf).
                    vaultName - The name of the vault.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
        glacier.close();
    }

    public static String uploadContent(GlacierClient glacier, Path path, String
 vaultName, File myFile) {
        // Get an SHA-256 tree hash value.
        String checkVal = computeSHA256(myFile);
        try {
            UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
                    .vaultName(vaultName)
                    .checksum(checkVal)
                    .build();

            UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
            return res.archiveId();

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    private static String computeSHA256(File inputFile) {
```

```
        try {
            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
            return toHex(treeHash);

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s", inputFile,
 ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
 %s", nsae.getMessage());
            System.exit(-1);
        }
        return "";
    }

    public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
            NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;
```

```
        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;

            while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
 file.getName(),
                            ioe.getMessage());
                }
            }
        }
    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     */
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[][] prevLvlHashes = chunkSHA256Hashes;
        while (prevLvlHashes.length > 1) {
            int len = prevLvlHashes.length / 2;
            if (prevLvlHashes.length % 2 != 0) {
                len++;
            }
```

```
            byte[][] currLvlHashes = new byte[len][];
            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

                // If there are at least two elements remaining.
                if (prevLvlHashes.length - i > 1) {

                    // Calculate a digest of the concatenated nodes.
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();

                } else { // Take care of the remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }

            prevLvlHashes = currLvlHashes;
        }

        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);
        for (byte datum : data) {
            String hex = Integer.toHexString(datum & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
}
```

- For API details, see UploadArchive in *AWS SDK for Java 2.x API Reference*.

SageMaker examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with SageMaker.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello SageMaker

The following code examples show how to get started using SageMaker.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSageMaker {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
```

```
        SageMakerClient sageMakerClient = SageMakerClient.builder()
                .region(region)
                .build();

        listBooks(sageMakerClient);
        sageMakerClient.close();
    }

    public static void listBooks(SageMakerClient sageMakerClient) {
        try {
            ListNotebookInstancesResponse notebookInstancesResponse =
 sageMakerClient.listNotebookInstances();
            List<NotebookInstanceSummary> items =
 notebookInstancesResponse.notebookInstances();
            for (NotebookInstanceSummary item : items) {
                System.out.println("The notebook name is: " +
 item.notebookInstanceName());
            }

        } catch (SageMakerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListNotebookInstances in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

CreatePipeline

The following code example shows how to use `CreatePipeline`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sagemaker#code-examples).

```
    // Create a pipeline from the example pipeline JSON.
    public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
            String functionArn, String pipelineName) {
        System.out.println("Setting up the pipeline.");
        JSONParser parser = new JSONParser();

        // Read JSON and get pipeline definition.
        try (FileReader reader = new FileReader(filePath)) {
            Object obj = parser.parse(reader);
            JSONObject jsonObject = (JSONObject) obj;
            JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
            for (Object stepObj : stepsArray) {
                JSONObject step = (JSONObject) stepObj;
                if (step.containsKey("FunctionArn")) {
                    step.put("FunctionArn", functionArn);
                }
            }
            System.out.println(jsonObject);

            // Create the pipeline.
            CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
                    .pipelineDescription("Java SDK example pipeline")
                    .roleArn(roleArn)
                    .pipelineName(pipelineName)
                    .pipelineDefinition(jsonObject.toString())
                    .build();

            sageMakerClient.createPipeline(pipelineRequest);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException | ParseException e) {
```

```
                throw new RuntimeException(e);
            }
        }
```

- For API details, see CreatePipeline in *AWS SDK for Java 2.x API Reference*.

DeletePipeline

The following code example shows how to use `DeletePipeline`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Delete a SageMaker pipeline by name.
    public static void deletePipeline(SageMakerClient sageMakerClient, String
 pipelineName) {
        DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
                .pipelineName(pipelineName)
                .build();

        sageMakerClient.deletePipeline(pipelineRequest);
        System.out.println("*** Successfully deleted " + pipelineName);
    }
```

- For API details, see DeletePipeline in *AWS SDK for Java 2.x API Reference*.

DescribePipelineExecution

The following code example shows how to use `DescribePipelineExecution`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Check the status of a pipeline execution.
    public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
            throws InterruptedException {
        String status;
        int index = 0;
        do {
            DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
                    .pipelineExecutionArn(executionArn)
                    .build();

            DescribePipelineExecutionResponse response = sageMakerClient
                    .describePipelineExecution(pipelineExecutionRequest);
            status = response.pipelineExecutionStatusAsString();
            System.out.println(index + ". The Status of the pipeline is " + status);
            TimeUnit.SECONDS.sleep(4);
            index++;
        } while ("Executing".equals(status));
        System.out.println("Pipeline finished with status " + status);
    }
```

- For API details, see DescribePipelineExecution in *AWS SDK for Java 2.x API Reference*.

StartPipelineExecution

The following code example shows how to use `StartPipelineExecution`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/usecases/workflow_sagemaker_pipes#code-examples).

```
    // Start a pipeline run with job configurations.
    public static String executePipeline(SageMakerClient sageMakerClient, String
 bucketName, String queueUrl,
            String roleArn, String pipelineName) {
        System.out.println("Starting pipeline execution.");
        String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
        String output = "s3://" + bucketName + "/outputfiles/";
        Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting().create();

        // Set up all parameters required to start the pipeline.
        List<Parameter> parameters = new ArrayList<>();
        Parameter para1 = Parameter.builder()
                .name("parameter_execution_role")
                .value(roleArn)
                .build();

        Parameter para2 = Parameter.builder()
                .name("parameter_queue_url")
                .value(queueUrl)
                .build();

        String inputJSON = "{\n" +
                "  \"DataSourceConfig\": {\n" +
                "    \"S3Data\": {\n" +
                "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
                "    },\n" +
                "    \"Type\": \"S3_DATA\"\n" +
                "  },\n" +
                "  \"DocumentType\": \"CSV\"\n" +
                "}";
```

```
        System.out.println(inputJSON);

        Parameter para3 = Parameter.builder()
                .name("parameter_vej_input_config")
                .value(inputJSON)
                .build();

        // Create an ExportVectorEnrichmentJobOutputConfig object.
        VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
                .s3Uri(output)
                .build();

        ExportVectorEnrichmentJobOutputConfig outputConfig =
 ExportVectorEnrichmentJobOutputConfig.builder()
                .s3Data(jobS3Data)
                .build();

        String gson4 = gson.toJson(outputConfig);
        Parameter para4 = Parameter.builder()
                .name("parameter_vej_export_config")
                .value(gson4)
                .build();
        System.out.println("parameter_vej_export_config:" +
 gson.toJson(outputConfig));

        // Create a VectorEnrichmentJobConfig object.
        ReverseGeocodingConfig reverseGeocodingConfig =
 ReverseGeocodingConfig.builder()
                .xAttributeName("Longitude")
                .yAttributeName("Latitude")
                .build();

        VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
                .reverseGeocodingConfig(reverseGeocodingConfig)
                .build();

        String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
        Parameter para5 = Parameter.builder()
                .name("parameter_step_1_vej_config")
                .value(para5JSON)
                .build();
```

```
        System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
        parameters.add(para1);
        parameters.add(para2);
        parameters.add(para3);
        parameters.add(para4);
        parameters.add(para5);

        StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
                .pipelineExecutionDescription("Created using Java SDK")
                .pipelineExecutionDisplayName(pipelineName + "-example-execution")
                .pipelineParameters(parameters)
                .pipelineName(pipelineName)
                .build();

        StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }
```

- For API details, see StartPipelineExecution in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with geospatial jobs and pipelines

The following code example shows how to:

- Set up resources for a pipeline.
- Set up a pipeline that executes a geospatial job.
- Start a pipeline execution.
- Monitor the status of the execution.
- View the output of the pipeline.
- Clean up resources.

For more information, see Create and run SageMaker pipelines using AWS SDKs on Community.aws.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public class SagemakerWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String eventSourceMapping = "";

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
 <functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
 <pipelineName>\n\n"
                +
                "Where:\n" +
                "    sageMakerRoleName - The name of the Amazon SageMaker role.\n\n"
 +
                "    lambdaRoleName - The name of the AWS Lambda role.\n\n" +
                "    functionFileLocation - The file location where the JAR file
 that represents the AWS Lambda function is located.\n\n"
                +
                "    functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).\n\n" +
                "    queueName - The name of the Amazon Simple Queue Service (Amazon
 SQS) queue.\n\n" +
                "    bucketName - The name of the Amazon Simple Storage Service
 (Amazon S3) bucket.\n\n" +
                "    lnglatData - The file location of the latlongtest.csv file
 required for this use case.\n\n" +
                "    spatialPipelinePath - The file location of the
 GeoSpatialPipeline.json file required for this use case.\n\n"
                +
                "    pipelineName - The name of the pipeline to create (for example,
 sagemaker-sdk-example-pipeline).\n\n";

        if (args.length != 9) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String sageMakerRoleName = args[0];
        String lambdaRoleName = args[1];
        String functionFileLocation = args[2];
        String functionName = args[3];
        String queueName = args[4];
        String bucketName = args[5];
        String lnglatData = args[6];
        String spatialPipelinePath = args[7];
        String pipelineName = args[8];
        String handlerName = "org.example.SageMakerLambdaFunction::handleRequest";

        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
                .region(region)
                .build();

        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        LambdaClient lambdaClient = LambdaClient.builder()
                .region(region)
                .build();

        SqsClient sqsClient = SqsClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon SageMaker pipeline example 
scenario.");
        System.out.println(
                "\nThis example workflow will guide you through setting up and 
running an" +
                        "\nAmazon SageMaker pipeline. The pipeline uses an AWS 
Lambda function and an" +
```

```
                        "\nAmazon SQS Queue. It runs a vector enrichment reverse
geocode job to" +
                        "\nreverse geocode addresses in an input file and store the
results in an export file.");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
       String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
       String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

       String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn,
               handlerName);
       String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
       System.out.println("The queue URL is " + queueUrl);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("Setting up bucket " + bucketName);
       if (!checkBucket(s3Client, bucketName)) {
           setupBucket(s3Client, bucketName);
           System.out.println("Put " + lnglatData + " into " + bucketName);
           putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
       }
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("Now we can create and run our pipeline.");
       setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
       String pipelineExecutionARN = executePipeline(sageMakerClient, bucketName,
queueUrl, sageMakerRoleArn,
               pipelineName);
       System.out.println("The pipeline execution ARN value is " +
pipelineExecutionARN);
       waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
       System.out.println("Getting output results " + bucketName);
       getOutputResults(s3Client, bucketName);
       System.out.println(DASHES);

       System.out.println(DASHES);
```

```
        System.out.println("The pipeline has completed. To view the pipeline and
 runs " +
                "in SageMaker Studio, follow these instructions:" +
                "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Do you want to delete the AWS resources used in this
 Workflow? (y/n)");
        Scanner in = new Scanner(System.in);
        String delResources = in.nextLine();
        if (delResources.compareTo("y") == 0) {
            System.out.println("Lets clean up the AWS resources. Wait 30 seconds");
            TimeUnit.SECONDS.sleep(30);
            deleteEventSourceMapping(lambdaClient);
            deleteSQSQueue(sqsClient, queueName);
            listBucketObjects(s3Client, bucketName);
            deleteBucket(s3Client, bucketName);
            deleteLambdaFunction(lambdaClient, functionName);
            deleteLambdaRole(iam, lambdaRoleName);
            deleteSagemakerRole(iam, sageMakerRoleName);
            deletePipeline(sageMakerClient, pipelineName);
        } else {
            System.out.println("The AWS Resources were not deleted!");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SageMaker pipeline scenario is complete.");
        System.out.println(DASHES);
    }

    private static void readObject(S3Client s3Client, String bucketName, String key)
 {
        System.out.println("Output file contents: \n");
        GetObjectRequest objectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        ResponseBytes<GetObjectResponse> objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
        byte[] byteArray = objectBytes.asByteArray();
```

```
        String text = new String(byteArray, StandardCharsets.UTF_8);
        System.out.println("Text output: " + text);
    }

    // Display some results from the output directory.
    public static void getOutputResults(S3Client s3Client, String bucketName) {
        System.out.println("Getting output results {bucketName}.");
        ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
                .bucket(bucketName)
                .prefix("outputfiles/")
                .build();

        ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
        List<S3Object> s3Objects = response.contents();
        for (S3Object object : s3Objects) {
            readObject(s3Client, bucketName, object.key());
        }
    }

    // Check the status of a pipeline execution.
    public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
 String executionArn)
            throws InterruptedException {
        String status;
        int index = 0;
        do {
            DescribePipelineExecutionRequest pipelineExecutionRequest =
 DescribePipelineExecutionRequest.builder()
                    .pipelineExecutionArn(executionArn)
                    .build();

            DescribePipelineExecutionResponse response = sageMakerClient
                    .describePipelineExecution(pipelineExecutionRequest);
            status = response.pipelineExecutionStatusAsString();
            System.out.println(index + ". The Status of the pipeline is " + status);
            TimeUnit.SECONDS.sleep(4);
            index++;
        } while ("Executing".equals(status));
        System.out.println("Pipeline finished with status " + status);
    }

    // Delete a SageMaker pipeline by name.
    public static void deletePipeline(SageMakerClient sageMakerClient, String
 pipelineName) {
```

```
        DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
                .pipelineName(pipelineName)
                .build();

        sageMakerClient.deletePipeline(pipelineRequest);
        System.out.println("*** Successfully deleted " + pipelineName);
    }

    // Create a pipeline from the example pipeline JSON.
    public static void setupPipeline(SageMakerClient sageMakerClient, String
 filePath, String roleArn,
            String functionArn, String pipelineName) {
        System.out.println("Setting up the pipeline.");
        JSONParser parser = new JSONParser();

        // Read JSON and get pipeline definition.
        try (FileReader reader = new FileReader(filePath)) {
            Object obj = parser.parse(reader);
            JSONObject jsonObject = (JSONObject) obj;
            JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
            for (Object stepObj : stepsArray) {
                JSONObject step = (JSONObject) stepObj;
                if (step.containsKey("FunctionArn")) {
                    step.put("FunctionArn", functionArn);
                }
            }
            System.out.println(jsonObject);

            // Create the pipeline.
            CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
                    .pipelineDescription("Java SDK example pipeline")
                    .roleArn(roleArn)
                    .pipelineName(pipelineName)
                    .pipelineDefinition(jsonObject.toString())
                    .build();

            sageMakerClient.createPipeline(pipelineRequest);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException | ParseException e) {
            throw new RuntimeException(e);
        }
```

```
    }

    // Start a pipeline run with job configurations.
    public static String executePipeline(SageMakerClient sageMakerClient, String
 bucketName, String queueUrl,
            String roleArn, String pipelineName) {
        System.out.println("Starting pipeline execution.");
        String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
        String output = "s3://" + bucketName + "/outputfiles/";
        Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting().create();

        // Set up all parameters required to start the pipeline.
        List<Parameter> parameters = new ArrayList<>();
        Parameter para1 = Parameter.builder()
                .name("parameter_execution_role")
                .value(roleArn)
                .build();

        Parameter para2 = Parameter.builder()
                .name("parameter_queue_url")
                .value(queueUrl)
                .build();

        String inputJSON = "{\n" +
                "  \"DataSourceConfig\": {\n" +
                "    \"S3Data\": {\n" +
                "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
                "    },\n" +
                "    \"Type\": \"S3_DATA\"\n" +
                "  },\n" +
                "  \"DocumentType\": \"CSV\"\n" +
                "}";

        System.out.println(inputJSON);

        Parameter para3 = Parameter.builder()
                .name("parameter_vej_input_config")
                .value(inputJSON)
                .build();
```

```
        // Create an ExportVectorEnrichmentJobOutputConfig object.
        VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
                .s3Uri(output)
                .build();

        ExportVectorEnrichmentJobOutputConfig outputConfig =
 ExportVectorEnrichmentJobOutputConfig.builder()
                .s3Data(jobS3Data)
                .build();

        String gson4 = gson.toJson(outputConfig);
        Parameter para4 = Parameter.builder()
                .name("parameter_vej_export_config")
                .value(gson4)
                .build();
        System.out.println("parameter_vej_export_config:" +
 gson.toJson(outputConfig));

        // Create a VectorEnrichmentJobConfig object.
        ReverseGeocodingConfig reverseGeocodingConfig =
 ReverseGeocodingConfig.builder()
                .xAttributeName("Longitude")
                .yAttributeName("Latitude")
                .build();

        VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
                .reverseGeocodingConfig(reverseGeocodingConfig)
                .build();

        String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
        Parameter para5 = Parameter.builder()
                .name("parameter_step_1_vej_config")
                .value(para5JSON)
                .build();

        System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
        parameters.add(para1);
        parameters.add(para2);
        parameters.add(para3);
        parameters.add(para4);
        parameters.add(para5);
```

```
        StartPipelineExecutionRequest pipelineExecutionRequest =
 StartPipelineExecutionRequest.builder()
                .pipelineExecutionDescription("Created using Java SDK")
                .pipelineExecutionDisplayName(pipelineName + "-example-execution")
                .pipelineParameters(parameters)
                .pipelineName(pipelineName)
                .build();

        StartPipelineExecutionResponse response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }

    public static void deleteEventSourceMapping(LambdaClient lambdaClient) {
        DeleteEventSourceMappingRequest eventSourceMappingRequest =
 DeleteEventSourceMappingRequest.builder()
                .uuid(eventSourceMapping)
                .build();

        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
    }

    public static void deleteSagemakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        try {
            for (String policy : sageMakerRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
 DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);
```

```
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        try {
            for (String policy : lambdaRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
 DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific AWS Lambda function.
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            awsLambda.deleteFunction(request);
            System.out.println("*** " + functionName + " was deleted");
```

```
        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Delete the specific S3 bucket.
    public static void deleteBucket(S3Client s3Client, String bucketName) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("*** " + bucketName + " was deleted.");
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                deleteBucketObjects(s3, bucketName, myValue.key());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName, String
 objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
                .key(objectName)
                .build());
        try {
            DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
```

```
                    .bucket(bucketName)
                    .delete(Delete.builder()
                            .objects(toDelete).build())
                    .build();

            s3.deleteObjects(dor);
            System.out.println("*** " + bucketName + " objects were deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific Amazon SQS queue.
    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
 String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key("samplefiles/" + objectKey)
                    .metadata(metadata)
                    .build();
```

```
            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
 + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void setupBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
 s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Set up the SQS queue to use with the pipeline.
    public static String setupQueue(SqsClient sqsClient, LambdaClient lambdaClient,
 String queueName,
            String lambdaName) {
        System.out.println("Setting up queue named " + queueName);
        try {
            Map<QueueAttributeName, String> queueAtt = new HashMap<>();
            queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
            queueAtt.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS, "5");
```

```
            queueAtt.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(queueAtt)
                    .build();

            sqsClient.createQueue(createQueueRequest);
            System.out.println("\nGet queue url");
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            TimeUnit.SECONDS.sleep(15);

            connectLambda(sqsClient, lambdaClient, getQueueUrlResponse.queueUrl(),
lambdaName);
            System.out.println("Queue ready with Url " +
getQueueUrlResponse.queueUrl());
            return getQueueUrlResponse.queueUrl();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    // Connect the queue to the Lambda function as an event source.
    public static void connectLambda(SqsClient sqsClient, LambdaClient lambdaClient,
String queueUrl,
            String lambdaName) {
        System.out.println("Connecting the Lambda function and queue for the
pipeline.");
        String queueArn = "";

        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);
        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributeNames(atts)
                .build();
```

```
        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
            System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
            queueArn = queueAtt.getValue();
        }

        CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
                .eventSourceArn(queueArn)
                .functionName(lambdaName)
                .build();

        CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
        eventSourceMapping = response1.uuid();
        System.out.println("The mapping between the event source and Lambda function
was successful");
    }

    // Create an AWS Lambda function.
    public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
            String handler) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
            FunctionCode code = FunctionCode.builder()
                    .zipFile(fileToUpload)
                    .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                    .functionName(functionName)
                    .description("SageMaker example function.")
                    .code(code)
                    .handler(handler)
                    .runtime(Runtime.JAVA11)
                    .timeout(200)
                    .memorySize(1024)
                    .role(role)
```

```
                    .build();

            // Create a Lambda function using a waiter.
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The function ARN is " +
functionResponse.functionArn());
            return functionResponse.functionArn();

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSageMakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        System.out.println("Creating a role to use with SageMaker.");
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
```

```
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : sageMakerRolePolicies) {
                AttachRolePolicyRequest attachRequest =
 AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    private static String createLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
```

```
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : lambdaRolePolicies) {
                AttachRolePolicyRequest attachRequest =
 AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());

        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    public static String checkFunction(LambdaClient lambdaClient, String
 functionName, String filePath, String role,
            String handler) {
        System.out.println("Create an AWS Lambda function used in this workflow.");
        String functionArn;
```

```
        try {
            // Does this function already exist.
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
            functionArn = response.configuration().functionArn();

        } catch (LambdaException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            functionArn = createLambdaFunction(lambdaClient, functionName, filePath,
role, handler);
        }
        return functionArn;
    }

    // Check to see if the specific S3 bucket exists. If the S3 bucket exists, this
    // method returns true.
    public static boolean checkBucket(S3Client s3, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3.headBucket(headBucketRequest);
            System.out.println(bucketName + " exists");
            return true;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Checks to see if the Amazon SQS queue exists. If not, this method creates a
    // new queue
    // and returns the ARN value.
    public static String checkQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
            String lambdaName) {
        System.out.println("Creating a queue for this use case.");
        String queueUrl;
```

```
        try {
            GetQueueUrlRequest request = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
            queueUrl = response.queueUrl();
            System.out.println(queueUrl);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            queueUrl = setupQueue(sqsClient, lambdaClient, queueName, lambdaName);
        }
        return queueUrl;
    }

    // Checks to see if the Lambda role exists. If not, this method creates it.
    public static String checkLambdaRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS Lambda to use.");
        String roleArn;
        try {
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createLambdaRole(iam, roleName);
        }
        return roleArn;
    }

    // Checks to see if the SageMaker role exists. If not, this method creates it.
    public static String checkSageMakerRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS SageMaker to use.");
        String roleArn;
        try {
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();
```

```
            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createSageMakerRole(iam, roleName);
        }
        return roleArn;
    }

    private static String[] getSageMakerRolePolicies() {
        String[] sageMakerRolePolicies = new String[3];
        sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/
AmazonSageMakerFullAccess";
        sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
 "AmazonSageMakerGeospatialFullAccess";
        sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        return sageMakerRolePolicies;
    }

    private static String[] getLambdaRolePolicies() {
        String[] lambdaRolePolicies = new String[5];
        lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
        lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess";
        lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
                + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
        lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole";
        return lambdaRolePolicies;
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution

- UpdatePipeline

Secrets Manager examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Secrets Manager.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

`GetSecretValue`

The following code example shows how to use `GetSecretValue`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-
secrets.html
 */
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <secretName>\s

                Where:
                    secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
                .region(region)
                .build();

        getValue(secretsClient, secretName);
        secretsClient.close();
    }
```

```
    public static void getValue(SecretsManagerClient secretsClient, String
 secretName) {
        try {
            GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                    .secretId(secretName)
                    .build();

            GetSecretValueResponse valueResponse =
 secretsClient.getSecretValue(valueRequest);
            String secret = valueResponse.secretString();
            System.out.println(secret);

        } catch (SecretsManagerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see GetSecretValue in *AWS SDK for Java 2.x API Reference*.

Amazon SES examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SES.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

ListIdentities

The following code example shows how to use `ListIdentities`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        listSESIdentities(client);
    }

    public static void listSESIdentities(SesClient client) {
        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
```

```
            List<String> identities = identitiesResponse.identities();
            for (String identity : identities) {
                System.out.println("The identity is " + identity);
            }

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListIdentities in *AWS SDK for Java 2.x API Reference*.

`ListTemplates`

The following code example shows how to use `ListTemplates`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;

public class ListTemplates {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        listAllTemplates(sesv2Client);
```

```
    }

    public static void listAllTemplates(SesV2Client sesv2Client) {
        try {
            ListEmailTemplatesRequest templatesRequest =
 ListEmailTemplatesRequest.builder()
                    .pageSize(1)
                    .build();

            ListEmailTemplatesResponse response =
 sesv2Client.listEmailTemplates(templatesRequest);
            response.templatesMetadata()
                    .forEach(template -> System.out.println("Template name: " +
 template.templateName()));

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListTemplates in *AWS SDK for Java 2.x API Reference*.

SendEmail

The following code example shows how to use `SendEmail`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
```

```
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageEmailRequest {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.\s
                    subject - The  subject line.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";
```

```
        try {
            send(client, sender, recipient, subject, bodyHTML);
            client.close();
            System.out.println("Done");

        } catch (MessagingException e) {
            e.getStackTrace();
        }
    }

    public static void send(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyHTML) throws MessagingException {

        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();

        Content sub = Content.builder()
                .data(subject)
                .build();

        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .message(msg)
                .source(sender)
                .build();
```

```
        try {
            System.out.println("Attempting to send an email through Amazon SES " +
 "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject> <fileLocation>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.\s
                    subject - The  subject line.\s
                    fileLocation - The location of a Microsoft Excel file to use as
 an attachment (C:/AWS/customers.xls).\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        String fileLocation = args[3];

        // The email body for recipients with non-HTML email clients.
        String bodyText = "Hello,\r\n" + "Please see the attached file for a list "
                + "of customers to contact.";

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
                + "<p>Please see the attached file for a " + "list of customers to
 contact.</p>" + "</body>"
                + "</html>";

        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        try {
            sendemailAttachment(client, sender, recipient, subject, bodyText,
 bodyHTML, fileLocation);
            client.close();
```

```
            System.out.println("Done");

        } catch (IOException | MessagingException e) {
            e.getStackTrace();
        }
    }

    public static void sendemailAttachment(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyText,
            String bodyHTML,
            String fileLocation) throws AddressException, MessagingException,
 IOException {

        java.io.File theFile = new java.io.File(fileLocation);
        byte[] fileContent = Files.readAllBytes(theFile.toPath());

        Session session = Session.getDefaultInstance(new Properties());

        // Create a new MimeMessage object.
        MimeMessage message = new MimeMessage(session);

        // Add subject, from and to lines.
        message.setSubject(subject, "UTF-8");
        message.setFrom(new InternetAddress(sender));
        message.setRecipients(Message.RecipientType.TO,
 InternetAddress.parse(recipient));

        // Create a multipart/alternative child container.
        MimeMultipart msgBody = new MimeMultipart("alternative");

        // Create a wrapper for the HTML and text parts.
        MimeBodyPart wrap = new MimeBodyPart();

        // Define the text part.
        MimeBodyPart textPart = new MimeBodyPart();
        textPart.setContent(bodyText, "text/plain; charset=UTF-8");

        // Define the HTML part.
        MimeBodyPart htmlPart = new MimeBodyPart();
        htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");
```

```
        // Add the text and HTML parts to the child container.
        msgBody.addBodyPart(textPart);
        msgBody.addBodyPart(htmlPart);

        // Add the child container to the wrapper object.
        wrap.setContent(msgBody);

        // Create a multipart/mixed parent container.
        MimeMultipart msg = new MimeMultipart("mixed");

        // Add the parent container to the message.
        message.setContent(msg);
        msg.addBodyPart(wrap);

        // Define the attachment.
        MimeBodyPart att = new MimeBodyPart();
        DataSource fds = new ByteArrayDataSource(fileContent,
                "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
        att.setDataHandler(new DataHandler(fds));

        String reportName = "WorkReport.xls";
        att.setFileName(reportName);

        // Add the attachment to the message.
        msg.addBodyPart(att);

        try {
            System.out.println("Attempting to send an email through Amazon SES " +
 "using the AWS SDK for Java...");

            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            message.writeTo(outputStream);

            ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

            byte[] arr = new byte[buf.remaining()];
            buf.get(arr);

            SdkBytes data = SdkBytes.fromByteArray(arr);
            RawMessage rawMessage = RawMessage.builder()
                    .data(data)
                    .build();
```

```
            SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
                    .rawMessage(rawMessage)
                    .build();

            client.sendRawEmail(rawEmailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Email sent using SesClient with attachment");
    }
}
```

- For API details, see SendEmail in *AWS SDK for Java 2.x API Reference*.

`SendTemplatedEmail`

The following code example shows how to use `SendTemplatedEmail`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Also, make sure that you create a template. See the following documentation
 * topic:
 *
 * https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
 */

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <template> <sender> <recipient>\s

                Where:
                    template - The name of the email template.
                    sender - An email address that represents the sender.\s
                    recipient - An email address that represents the recipient.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String templateName = args[0];
        String sender = args[1];
        String recipient = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        send(sesv2Client, sender, recipient, templateName);
    }

    public static void send(SesV2Client client, String sender, String recipient,
 String templateName) {
        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();
```

```
        /*
         * Specify both name and favorite animal (favoriteanimal) in your code when
         * defining the Template object.
         * If you don't specify all the variables in the template, Amazon SES
 doesn't
         * send the email.
         */
        Template myTemplate = Template.builder()
                .templateName(templateName)
                .templateData("{\n" +
                        "  \"name\": \"Jason\"\n," +
                        "  \"favoriteanimal\": \"Cat\"\n" +
                        "}")
                .build();

        EmailContent emailContent = EmailContent.builder()
                .template(myTemplate)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

        try {
            System.out.println("Attempting to send an email based on a template
 using the AWS SDK for Java (v2)...");
            client.sendEmail(emailRequest);
            System.out.println("email based on a template was sent");

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see SendTemplatedEmail in *AWS SDK for Java 2.x API Reference*.

Amazon SES API v2 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SES API v2.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

`CreateContact`

The following code example shows how to use `CreateContact`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    try {
      // Create a new contact with the provided email address in the
      CreateContactRequest contactRequest = CreateContactRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .emailAddress(emailAddress)
          .build();
```

```
        sesClient.createContact(contactRequest);
        contacts.add(emailAddress);

        System.out.println("Contact created: " + emailAddress);

        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
                            .html(Content.builder().data(welcomeHtml).build())
                            .build())
                        .build())
                .build())
            .build();
        SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
      } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
 proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
      } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
        throw e;
      }
    }
```

- For API details, see CreateContact in *AWS SDK for Java 2.x API Reference*.

CreateContactList

The following code example shows how to use `CreateContactList`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
 CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
      throw e;
    }
```

- For API details, see CreateContactList in *AWS SDK for Java 2.x API Reference*.

CreateEmailIdentity

The following code example shows how to use `CreateEmailIdentity`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ses#code-examples).

```
    try {
      CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
          .emailIdentity(verifiedEmail)
          .build();
      sesClient.createEmailIdentity(createEmailIdentityRequest);
      System.out.println("Email identity created: " + verifiedEmail);
    } catch (AlreadyExistsException e) {
      System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
    } catch (NotFoundException e) {
      System.err.println("The provided email address is not verified: " +
verifiedEmail);
      throw e;
    } catch (LimitExceededException e) {
      System.err
          .println("You have reached the limit for email identities. Please remove
some identities and try again.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating email identity: " + e.getMessage());
      throw e;
    }
```

- For API details, see CreateEmailIdentity in *AWS SDK for Java 2.x API Reference*.

`CreateEmailTemplate`

The following code example shows how to use `CreateEmailTemplate`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/usecases/workflow_ses).

```
    try {
      // Create an email template named "weekly-coupons"
      String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
      String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

      CreateEmailTemplateRequest templateRequest =
 CreateEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .templateContent(EmailTemplateContent.builder()
              .subject("Weekly Coupons Newsletter")
              .html(newsletterHtml)
              .text(newsletterText)
              .build())
          .build();

      sesClient.createEmailTemplate(templateRequest);

      System.out.println("Email template created: " + TEMPLATE_NAME);
    } catch (AlreadyExistsException e) {
      // If the template already exists, skip this step and proceed with the next
      // operation
      System.out.println("Email template already exists, skipping creation...");
    } catch (LimitExceededException e) {
      // If the limit for email templates is exceeded, fail the workflow and inform
      // the user
      System.err.println("You have reached the limit for email templates. Please
 remove some templates and try again.");
      throw e;
    } catch (Exception e) {
      System.err.println("Error occurred while creating email template: " +
 e.getMessage());
      throw e;
```

```
    }
```

- For API details, see CreateEmailTemplate in *AWS SDK for Java 2.x API Reference*.

DeleteContactList

The following code example shows how to use `DeleteContactList`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    try {
      // Delete the contact list
      DeleteContactListRequest deleteContactListRequest =
 DeleteContactListRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      sesClient.deleteContactList(deleteContactListRequest);

      System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
    } catch (NotFoundException e) {
      // If the contact list does not exist, log the error and proceed
      System.out.println("Contact list not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the contact list: " +
 e.getMessage());
      e.printStackTrace();
    }
```

- For API details, see DeleteContactList in *AWS SDK for Java 2.x API Reference*.

`DeleteEmailIdentity`

The following code example shows how to use `DeleteEmailIdentity`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sesv2#code-examples).

```
      try {
        // Delete the email identity
        DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
            .emailIdentity(this.verifiedEmail)
            .build();

        sesClient.deleteEmailIdentity(deleteIdentityRequest);

        System.out.println("Email identity deleted: " + this.verifiedEmail);
      } catch (NotFoundException e) {
        // If the email identity does not exist, log the error and proceed
        System.out.println("Email identity not found. Skipping deletion...");
      } catch (Exception e) {
        System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
        e.printStackTrace();
      }
    } else {
      System.out.println("Skipping email identity deletion.");
    }
```

- For API details, see [DeleteEmailIdentity](https://docs.aws.amazon.com/goto/SdkForJavaV2/sesv2-2019-09-27/DeleteEmailIdentity) in *AWS SDK for Java 2.x API Reference*.

`DeleteEmailTemplate`

The following code example shows how to use `DeleteEmailTemplate`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ses#code-examples).

```
    try {
      // Delete the template
      DeleteEmailTemplateRequest deleteTemplateRequest =
 DeleteEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .build();

      sesClient.deleteEmailTemplate(deleteTemplateRequest);

      System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
      // If the email template does not exist, log the error and proceed
      System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email template: " +
 e.getMessage());
      e.printStackTrace();
    }
```

- For API details, see [DeleteEmailTemplate](https://docs.aws.amazon.com/goto/SdkForJavaV2/sesv2-2019-09-27/DeleteEmailTemplate) in *AWS SDK for Java 2.x API Reference*.

`ListContacts`

The following code example shows how to use `ListContacts`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ses#code-examples).

```
        ListContactsRequest contactListRequest = ListContactsRequest.builder()
            .contactListName(CONTACT_LIST_NAME)
            .build();

        List<String> contactEmails;
        try {
          ListContactsResponse contactListResponse =
 sesClient.listContacts(contactListRequest);

          contactEmails = contactListResponse.contacts().stream()
              .map(Contact::emailAddress)
              .toList();
        } catch (Exception e) {
          // TODO: Remove when listContacts's GET body issue is resolved.
          contactEmails = this.contacts;
        }
```

- For API details, see ListContacts in *AWS SDK for Java 2.x API Reference*.

SendEmail

The following code example shows how to use SendEmail.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Sends a message.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
```

```
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendEmail {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                    <sender> <recipient> <subject>\s

                                Where:
                                    sender - An email address that represents the
 sender.\s
                                    recipient - An email address that represents the
 recipient.\s
                                    subject - The subject line.\s
                                """;

                if (args.length != 3) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String sender = args[0];
                String recipient = args[1];
                String subject = args[2];

                Region region = Region.US_EAST_1;
                SesV2Client sesv2Client = SesV2Client.builder()
                                .region(region)
                                .build();

                // The HTML body of the email.
```

```
                String bodyHTML = "<html>" + "<head></head>" + "<body>" +
"<h1>Hello!</h1>"
                                + "<p> See the list of customers.</p>" + "</body>" +
"</html>";

                send(sesv2Client, sender, recipient, subject, bodyHTML);
        }

        public static void send(SesV2Client client,
                        String sender,
                        String recipient,
                        String subject,
                        String bodyHTML) {

                Destination destination = Destination.builder()
                                .toAddresses(recipient)
                                .build();

                Content content = Content.builder()
                                .data(bodyHTML)
                                .build();

                Content sub = Content.builder()
                                .data(subject)
                                .build();

                Body body = Body.builder()
                                .html(content)
                                .build();

                Message msg = Message.builder()
                                .subject(sub)
                                .body(body)
                                .build();

                EmailContent emailContent = EmailContent.builder()
                                .simple(msg)
                                .build();

                SendEmailRequest emailRequest = SendEmailRequest.builder()
                                .destination(destination)
                                .content(emailContent)
                                .fromEmailAddress(sender)
                                .build();
```

```
                try {
                        System.out.println("Attempting to send an email through
 Amazon SES "
                                        + "using the AWS SDK for Java...");
                        client.sendEmail(emailRequest);
                        System.out.println("email was sent");

                } catch (SesV2Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

Sends a message using a template.

```
      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
                    .templateName(TEMPLATE_NAME)
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
      }
```

- For API details, see SendEmail in *AWS SDK for Java 2.x API Reference*.

Scenarios

Newsletter workflow

The following code example shows how to Amazon SES API v2 newsletter workflow.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ses#code-examples).

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
 CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
      throw e;
    }

      try {
        // Create a new contact with the provided email address in the
        CreateContactRequest contactRequest = CreateContactRequest.builder()
            .contactListName(CONTACT_LIST_NAME)
            .emailAddress(emailAddress)
            .build();

        sesClient.createContact(contactRequest);
        contacts.add(emailAddress);
```

```
        System.out.println("Contact created: " + emailAddress);

        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
                            .html(Content.builder().data(welcomeHtml).build())
                            .build())
                        .build())
                .build())
            .build();
        SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
      } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
 proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
      } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
        throw e;
      }
    }

      ListContactsRequest contactListRequest = ListContactsRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();
```

```
      List<String> contactEmails;
      try {
        ListContactsResponse contactListResponse =
 sesClient.listContacts(contactListRequest);

        contactEmails = contactListResponse.contacts().stream()
            .map(Contact::emailAddress)
            .toList();
      } catch (Exception e) {
        // TODO: Remove when listContacts's GET body issue is resolved.
        contactEmails = this.contacts;
      }


      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
                    .templateName(TEMPLATE_NAME)
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
      }

    try {
      CreateEmailIdentityRequest createEmailIdentityRequest =
 CreateEmailIdentityRequest.builder()
          .emailIdentity(verifiedEmail)
          .build();
      sesClient.createEmailIdentity(createEmailIdentityRequest);
      System.out.println("Email identity created: " + verifiedEmail);
    } catch (AlreadyExistsException e) {
```

```
      System.out.println("Email identity already exists, skipping creation: " +
 verifiedEmail);
    } catch (NotFoundException e) {
      System.err.println("The provided email address is not verified: " +
 verifiedEmail);
      throw e;
    } catch (LimitExceededException e) {
      System.err
          .println("You have reached the limit for email identities. Please remove
 some identities and try again.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating email identity: " + e.getMessage());
      throw e;
    }

    try {
      // Create an email template named "weekly-coupons"
      String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
      String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

      CreateEmailTemplateRequest templateRequest =
 CreateEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .templateContent(EmailTemplateContent.builder()
              .subject("Weekly Coupons Newsletter")
              .html(newsletterHtml)
              .text(newsletterText)
              .build())
          .build();

      sesClient.createEmailTemplate(templateRequest);

      System.out.println("Email template created: " + TEMPLATE_NAME);
    } catch (AlreadyExistsException e) {
      // If the template already exists, skip this step and proceed with the next
      // operation
      System.out.println("Email template already exists, skipping creation...");
    } catch (LimitExceededException e) {
      // If the limit for email templates is exceeded, fail the workflow and inform
      // the user
```

```
      System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
      throw e;
    } catch (Exception e) {
      System.err.println("Error occurred while creating email template: " +
e.getMessage());
      throw e;
    }

    try {
      // Delete the contact list
      DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      sesClient.deleteContactList(deleteContactListRequest);

      System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
    } catch (NotFoundException e) {
      // If the contact list does not exist, log the error and proceed
      System.out.println("Contact list not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
      e.printStackTrace();
    }

      try {
        // Delete the email identity
        DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
            .emailIdentity(this.verifiedEmail)
            .build();

        sesClient.deleteEmailIdentity(deleteIdentityRequest);

        System.out.println("Email identity deleted: " + this.verifiedEmail);
      } catch (NotFoundException e) {
        // If the email identity does not exist, log the error and proceed
        System.out.println("Email identity not found. Skipping deletion...");
      } catch (Exception e) {
        System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
```

```
        e.printStackTrace();
      }
    } else {
      System.out.println("Skipping email identity deletion.");
    }

    try {
      // Delete the template
      DeleteEmailTemplateRequest deleteTemplateRequest =
 DeleteEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .build();

      sesClient.deleteEmailTemplate(deleteTemplateRequest);

      System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
      // If the email template does not exist, log the error and proceed
      System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email template: " +
 e.getMessage());
      e.printStackTrace();
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Amazon SNS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SNS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon SNS

The following code examples show how to get started using Amazon SNS.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListTopics in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios
- Serverless examples

Actions

CheckIfPhoneNumberIsOptedOut

The following code example shows how to use `CheckIfPhoneNumberIsOptedOut`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }
```

```
    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see CheckIfPhoneNumberIsOptedOut in *AWS SDK for Java 2.x API Reference*.

`ConfirmSubscription`

The following code example shows how to use `ConfirmSubscription`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionToken> <topicArn>

                Where:
                   subscriptionToken - A short-lived token sent to an endpoint
 during the Subscribe action.
                   topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        confirmSub(snsClient, subscriptionToken, topicArn);
        snsClient.close();
    }

    public static void confirmSub(SnsClient snsClient, String subscriptionToken,
 String topicArn) {
        try {
            ConfirmSubscriptionRequest request =
 ConfirmSubscriptionRequest.builder()
                    .token(subscriptionToken)
                    .topicArn(topicArn)
                    .build();
```

```
            ConfirmSubscriptionResponse result =
 snsClient.confirmSubscription(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
                    + result.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ConfirmSubscription in *AWS SDK for Java 2.x API Reference*.

`CreateTopic`

The following code example shows how to use `CreateTopic`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
```

```
    }
}
```

- For API details, see CreateTopic in *AWS SDK for Java 2.x API Reference*.

DeleteTopic

The following code example shows how to use `DeleteTopic`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteTopic in *AWS SDK for Java 2.x API Reference*.

GetSMSAttributes

The following code example shows how to use `GetSMSAttributes`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetSMSAtrributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic from which to retrieve
 attributes.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
 GetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
 snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
 entry.getValue());
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("\n\nStatus was good");
    }
}
```

- For API details, see GetSMSAttributes in *AWS SDK for Java 2.x API Reference*.

GetTopicAttributes

The following code example shows how to use `GetTopicAttributes`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sns#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to look up.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        System.out.println("Getting attributes for a topic with name: " + topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn) {
        try {
            GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()
                    .topicArn(topicArn)
                    .build();

            GetTopicAttributesResponse result =
 snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " +
 result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                    + result.attributes());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see GetTopicAttributes in *AWS SDK for Java 2.x API Reference*.

`ListPhoneNumbersOptedOut`

The following code example shows how to use `ListPhoneNumbersOptedOut`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " + result.sdkHttpResponse().statusCode()
 + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListPhoneNumbersOptedOut in *AWS SDK for Java 2.x API Reference*.

ListSubscriptions

The following code example shows how to use `ListSubscriptions`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sns#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                    .build();

            ListSubscriptionsResponse result = snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());
```

```
        } catch (SnsException e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListSubscriptions in *AWS SDK for Java 2.x API Reference*.

`ListTopics`

The following code example shows how to use `ListTopics`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
```

```
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                    .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                    "Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListTopics in *AWS SDK for Java 2.x API Reference*.

Publish

The following code example shows how to use `Publish`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
```

```
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see Publish in *AWS SDK for Java 2.x API Reference*.

SetSMSAttributes

The following code example shows how to use `SetSMSAttributes`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
 attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
 was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see SetSMSAttributes in *AWS SDK for Java 2.x API Reference*.

SetSubscriptionAttributes

The following code example shows how to use `SetSubscriptionAttributes`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sns#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of a subscription.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");

            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
            attributeValues.add("rugby");
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);

            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [SetSubscriptionAttributes](#) in *AWS SDK for Java 2.x API Reference*.

SetTopicAttributes

The following code example shows how to use SetTopicAttributes.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sns#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
            SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
                    .topicArn(topicArn)
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() + "\n
\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see SetTopicAttributes in *AWS SDK for Java 2.x API Reference*.

Subscribe

The following code example shows how to use `Subscribe`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Subscribe an email address to a topic.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = """
                Usage:     <topicArn> <email>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   email - The email address to use.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String email)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Subscribe an HTTP endpoint to a topic.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <url>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   url - The HTTPS endpoint that you want to receive notifications.
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("https")
                    .endpoint(url)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn() +
 "\n\n Status is "
```

```
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Subscribe a Lambda function to a topic.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = """

                Usage:    <topicArn> <lambdaArn>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   lambdaArn - The ARN of an AWS Lambda function.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
 lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("lambda")
                    .endpoint(lambdaArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see Subscribe in *AWS SDK for Java 2.x API Reference*.

TagResource

The following code example shows how to use `TagResource`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sns#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to which tags are added.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                    .key("Team")
                    .value("Development")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Environment")
                    .value("Gamma")
                    .build();

            List<Tag> tagList = new ArrayList<>();
            tagList.add(tag);
            tagList.add(tag2);

            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(topicArn)
                    .tags(tagList)
                    .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see TagResource in *AWS SDK for Java 2.x API Reference*.

Unsubscribe

The following code example shows how to use `Unsubscribe`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sns#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see Unsubscribe in *AWS SDK for Java 2.x API Reference*.

Scenarios

Create a platform endpoint for push notifications

The following code example shows how to create a platform endpoint for Amazon SNS push notifications.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform application.
 You can get this value from the AWS Management Console.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Create and publish to a FIFO topic

The following code example shows how to create and publish to a FIFO Amazon SNS topic.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

This example

- creates an Amazon SNS FIFO topic, two Amazon SQS FIFO queues, and one Standard queue.
- subscribes the queues to the topic and publishes a message to the topic.

The test verifies the receipt of the message to each queue. The complete example also shows the addition of access policies and deletes the resources at the end.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
                "Where:\n" +
                "   fifoTopicName - The name of the FIFO topic that you want to
 create. \n\n" +
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
 created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will created
 for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that will
 be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
```

```
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue: ARN,
 URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
 "Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
                    "ContentBasedDeduplication", "false");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();
```

```
            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                    .topicArn(topicARN)
                    .endpoint(queue.queueARN)
                    .protocol("sqs")
                    .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon SNS
FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";
```

```
            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue(attributeValue)
                    .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                    .topicArn(topicArn)
                    .subject(subject)
                    .message(payload)
                    .messageGroupId(groupId)
                    .messageDeduplicationId(dedupId)
                    .messageAttributes(attributes)
                    .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [CreateTopic]
  - [Publish]
  - [Subscribe]

Publish SMS messages to a topic

The following code example shows how to:

- Create an Amazon SNS topic.
- Subscribe phone numbers to the topic.
- Publish SMS messages to the topic so that all subscribed phone numbers receive the message at once.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a topic and return its ARN.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

Subscribe an endpoint to a topic.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <phoneNumber>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   phoneNumber - A mobile phone number that receives notifications
 (for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Set attributes on the message, such as the ID of the sender, the maximum price, and its type. Message attributes are optional.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
 attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
```

```
                    .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
 was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Publish a message to a topic. The message is sent to every subscriber.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is sent
 (for example, +1XXX5550100).\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is sent
 (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see Publish in *AWS SDK for Java 2.x API Reference*.

Publish messages to queues

The following code example shows how to:

- Create topic (FIFO or non-FIFO).
- Subscribe several queues to the topic with an option to apply a filter.
- Publish messages to the topic.
- Poll the queues for messages received.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sns#code-examples).

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.ArrayList;
```

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
                "Usage:\n" +
                "    <fifoQueueARN>\n\n" +
                "Where:\n" +
                "    accountId - Your AWS account Id value.";

        // if (args.length != 1) {
        // System.out.println(usage);
        // System.exit(1);
```

```
        // }

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();

        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();

        Scanner in = new Scanner(System.in);
        String accountId = "814548047983";
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this workflow, you will create an SNS topic and
subscribe an SQS queue to the topic.\n" +
                "You can select from several options for configuring the topic and
the subscriptions for the queue.\n" +
                "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
                "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
                "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic, deduplication
 is supported.\n" +
                    "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                    +
                    "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,\n"
                    +
                    "        within the five-minute deduplication interval, is
 accepted but not delivered.\n" +
                    "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                    "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
```

```
            System.out.println("Because you have selected a FIFO topic, '.fifo' must
be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you are
        // running this code
        // in a different region.
        String policy = "{\n" +
                "     \"Statement\": [\n" +
                "     {\n" +
                "         \"Effect\": \"Allow\",\n" +
                "                 \"Principal\": {\n" +
                "             \"Service\": \"sns.amazonaws.com\"\n" +
```

```
                "        },\n" +
                "        \"Action\": \"sqs:SendMessage\",\n" +
                "                \"Resource\": \"arn:aws:sqs:us-east-1:" +
 accountId + ":" + sqsQueueName + "\",\n" +
                "                \"Condition\": {\n" +
                "            \"ArnEquals\": {\n" +
                "                \"aws:SourceArn\": \"arn:aws:sns:us-east-1:" +
 accountId + ":" + topicName + "\"\n" +
                "            }\n" +
                "        }\n" +
                "    }\n" +
                "    ]\n" +
                " }";

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                    "If you add a filter to this subscription, then only the
 filtered messages will be received in the queue.\n"
                            +
                            "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                            +
                            "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                    + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
```

```
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this message?
 (y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
```

```
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
 groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
 continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
 Press any key to continue.");
        in.nextLine();
```

```
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode()
                    + "\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
 List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
 DeleteMessageBatchRequestEntry.builder()
                        .id(msg.messageId())
                        .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
 DeleteMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .maxNumberOfMessages(5)
                        .build();
                return sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                        .maxNumberOfMessages(5)
                        .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();
```

```
            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
            String message,
            String topicArn,
            String msgAttValue,
            String duplication,
            String groupId,
            String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                } else {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageDeduplicationId(deduplicationID)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
 HashMap<>();
                messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
                        .dataType("String")
```

```
                        .stringValue("true")
                        .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                            .topicArn(topicArn)
                            .message(message)
                            .messageDeduplicationId(deduplicationID)
                            .messageGroupId(groupId)
                            .messageAttributes(messageAttributes)
                            .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                        .protocol("sqs")
                        .endpoint(queueArn)
                        .returnSubscriptionArn(true)
```

```
                        .topicArn(topicArn)
                        .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
 to the topic " + topicArn + "\n" +
                        "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                        .protocol("sqs")
                        .endpoint(queueArn)
                        .returnSubscriptionArn(true)
                        .topicArn(topicArn)
                        .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
 to the topic " + topicArn + "\n" +
                        "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
 SetSubscriptionAttributesRequest.builder()
                        .subscriptionArn(result.subscriptionArn())
                        .attributeName(attributeName)
                        .attributeValue(updatedJsonString)
                        .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
 policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
 String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);

            SetQueueAttributesRequest attributesRequest =
 SetQueueAttributesRequest.builder()
                    .queueUrl(queueUrl)
                    .attributes(attrMap)
                    .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
 GetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributeNames(atts)
                .build();

        GetQueueAttributesResponse response =
 sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
```

```
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName, Boolean
 selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                        .queueName(queueName)
                        .attributes(attrs)
                        .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

 .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                        .queueName(queueName)
                        .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

 .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
```

```
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Serverless examples

Invoke a Lambda function from an Amazon SNS trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving messages from an SNS topic. The function retrieves the messages from the event parameter and logs the content of each message.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an SNS event with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;
```

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

Amazon SQS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SQS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon SQS

The following code examples show how to get started using Amazon SQS.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.awssdk.services.sqs.paginators.ListQueuesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSQS {
    public static void main(String[] args) {
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        listQueues(sqsClient);
        sqsClient.close();
    }

    public static void listQueues(SqsClient sqsClient) {
        try {
            ListQueuesIterable listQueues = sqsClient.listQueuesPaginator();
            listQueues.stream()
                    .flatMap(r -> r.queueUrls().stream())
                    .forEach(content -> System.out.println(" Queue URL: " +
 content.toLowerCase()));

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [ListQueues]() in *AWS SDK for Java 2.x API Reference*.

Topics

- [Actions]()
- [Scenarios]()
- [Serverless examples]()

Actions

`CreateQueue`

The following code example shows how to use `CreateQueue`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sqs#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
```

```
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");

            GetQueueUrlResponse getQueueUrlResponse = sqsClient

 .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void listQueues(SqsClient sqsClient) {

        System.out.println("\nList Queues");
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
 ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
 sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
```

```
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
        // List queues with filters
        String namePrefix = "queue";
        ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
                .queueNamePrefix(namePrefix)
                .build();

        ListQueuesResponse listQueuesFilteredResponse = 
 sqsClient.listQueues(filterListRequest);
        System.out.println("Queue URLs with prefix: " + namePrefix);
        for (String url : listQueuesFilteredResponse.queueUrls()) {
            System.out.println(url);
        }

        System.out.println("\nSend message");
        try {
            sqsClient.sendMessage(SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody("Hello world!")
                    .delaySeconds(10)
                    .build());

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

        System.out.println("\nSend multiple messages");
        try {
            SendMessageBatchRequest sendMessageBatchRequest = 
 SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
```

```
.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from 
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg 
2").delaySeconds(10)
                                    .build())
                    .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String 
queueUrl) {

        System.out.println("\nReceive messages");
        try {
            ReceiveMessageRequest receiveMessageRequest = 
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void changeMessages(SqsClient sqsClient, String queueUrl, 
List<Message> messages) {

        System.out.println("\nChange Message Visibility");
        try {

            for (Message message : messages) {
                ChangeMessageVisibilityRequest req = 
ChangeMessageVisibilityRequest.builder()
```

```
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .visibilityTimeout(100)
                        .build();
                sqsClient.changeMessageVisibility(req);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
 List<Message> messages) {
        System.out.println("\nDelete Messages");

        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
 DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see CreateQueue in *AWS SDK for Java 2.x API Reference*.

DeleteMessage

The following code example shows how to use `DeleteMessage`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

- For API details, see DeleteMessage in *AWS SDK for Java 2.x API Reference*.

`DeleteQueue`

The following code example shows how to use `DeleteQueue`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
```

```
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteQueue {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <queueName>

                Where:
                   queueName - The name of the Amazon SQS queue to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        SqsClient sqs = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        deleteSQSQueue(sqs, queueName);
        sqs.close();
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
```

```
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteQueue in *AWS SDK for Java 2.x API Reference*.

GetQueueUrl

The following code example shows how to use `GetQueueUrl`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

  .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
```

- For API details, see GetQueueUrl in *AWS SDK for Java 2.x API Reference*.

ListQueues

The following code example shows how to use `ListQueues`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

- For API details, see ListQueues in *AWS SDK for Java 2.x API Reference*.

`ReceiveMessage`

The following code example shows how to use `ReceiveMessage`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        try {
```

```
            ReceiveMessageRequest receiveMessageRequest =
 ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
```

- For API details, see ReceiveMessage in *AWS SDK for Java 2.x API Reference*.

SendMessage

The following code example shows how to use `SendMessage`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessages {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <queueName> <message>

                Where:
                   queueName - The name of the queue.
                   message - The message to send.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        String message = args[1];
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();
        sendMessage(sqsClient, queueName, message);
        sqsClient.close();
    }

    public static void sendMessage(SqsClient sqsClient, String queueName, String
 message) {
        try {
            CreateQueueRequest request = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();
            sqsClient.createQueue(request);

            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(message)
                    .delaySeconds(5)
```

```
                    .build();

            sqsClient.sendMessage(sendMsgRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see SendMessage in *AWS SDK for Java 2.x API Reference*.

SendMessageBatch

The following code example shows how to use SendMessageBatch.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
                                .build())
                    .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- For API details, see SendMessageBatch in *AWS SDK for Java 2.x API Reference*.

Scenarios

Create and publish to a FIFO topic

The following code example shows how to create and publish to a FIFO Amazon SNS topic.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

This example

- creates an Amazon SNS FIFO topic, two Amazon SQS FIFO queues, and one Standard queue.
- subscribes the queues to the topic and publishes a message to the topic.

The test verifies the receipt of the message to each queue. The complete example also shows the addition of access policies and deletes the resources at the end.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
                "Where:\n" +
                "   fifoTopicName - The name of the FIFO topic that you want to
 create. \n\n" +
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
 created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will created
 for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that will
 be created for the analytics consumer. \n\n";
        if (args.length != 4) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue: ARN,
 URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
 "Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
```

```
                    "ContentBasedDeduplication", "false");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                    .topicArn(topicARN)
                    .endpoint(queue.queueARN)
                    .protocol("sqs")
                    .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon SNS
FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
```

```
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue(attributeValue)
                    .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                    .topicArn(topicArn)
                    .subject(subject)
                    .message(payload)
                    .messageGroupId(groupId)
                    .messageDeduplicationId(dedupId)
                    .messageAttributes(attributes)
                    .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateTopic
  - Publish
  - Subscribe

Publish messages to queues

The following code example shows how to:

- Create topic (FIFO or non-FIFO).
- Subscribe several queues to the topic with an option to apply a filter.
- Publish messages to the topic.
- Poll the queues for messages received.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
```

```
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
                "Usage:\n" +
```

```
                "    <fifoQueueARN>\n\n" +
                "Where:\n" +
                "    accountId - Your AWS account Id value.";

        // if (args.length != 1) {
        // System.out.println(usage);
        // System.exit(1);
        // }

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();

        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();

        Scanner in = new Scanner(System.in);
        String accountId = "814548047983";
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
```

```
        System.out.println("In this workflow, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
                "You can select from several options for configuring the topic and
 the subscriptions for the queue.\n" +
                "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
                "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
                "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic, deduplication
 is supported.\n" +
                    "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                    +
                    "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,\n"
                    +
                    "        within the five-minute deduplication interval, is
 accepted but not delivered.\n" +
                    "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                    "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo' must
 be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you are
        // running this code
```

```
        // in a different region.
        String policy = "{\n" +
                "     \"Statement\": [\n" +
                "     {\n" +
                "         \"Effect\": \"Allow\",\n" +
                "                 \"Principal\": {\n" +
                "             \"Service\": \"sns.amazonaws.com\"\n" +
                "         },\n" +
                "         \"Action\": \"sqs:SendMessage\",\n" +
                "                 \"Resource\": \"arn:aws:sqs:us-east-1:" +
 accountId + ":" + sqsQueueName + "\",\n" +
                "                 \"Condition\": {\n" +
                "             \"ArnEquals\": {\n" +
                "                 \"aws:SourceArn\": \"arn:aws:sns:us-east-1:" +
 accountId + ":" + topicName + "\"\n" +
                "             }\n" +
                "         }\n" +
                "     }\n" +
                "     ]\n" +
                " }";

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                    "If you add a filter to this subscription, then only the
 filtered messages will be received in the queue.\n"
                            +
                            "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                            +
                            "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                    + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
```

```
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this message?
(y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
```

```
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
 groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
 continue.");
        in.nextLine();
```

```
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
 Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
```

```
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode()
                    + "\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
 List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
 DeleteMessageBatchRequestEntry.builder()
                        .id(msg.messageId())
                        .build();

                entries.add(entry);
            }
```

```
            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .maxNumberOfMessages(5)
                        .build();
                return sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                        .maxNumberOfMessages(5)
                        .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

```
    public static void pubMessage(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
            String message,
            String topicArn,
            String msgAttValue,
            String duplication,
            String groupId,
            String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                } else {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageDeduplicationId(deduplicationID)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
```

```
                            .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
 HashMap<>();
                messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
                        .dataType("String")
                        .stringValue("true")
                        .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                            .topicArn(topicArn)
                            .message(message)
                            .messageDeduplicationId(deduplicationID)
                            .messageGroupId(groupId)
                            .messageAttributes(messageAttributes)
                            .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
 queueArn, List<String> filterList) {
```

```
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                        .protocol("sqs")
                        .endpoint(queueArn)
                        .returnSubscriptionArn(true)
                        .topicArn(topicArn)
                        .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
 to the topic " + topicArn + "\n" +
                        "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                        .protocol("sqs")
                        .endpoint(queueArn)
                        .returnSubscriptionArn(true)
                        .topicArn(topicArn)
                        .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
 to the topic " + topicArn + "\n" +
                        "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
 SetSubscriptionAttributesRequest.builder()
                        .subscriptionArn(result.subscriptionArn())
                        .attributeName(attributeName)
```

```
                        .attributeValue(updatedJsonString)
                        .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
 policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
 String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);

            SetQueueAttributesRequest attributesRequest =
 SetQueueAttributesRequest.builder()
                    .queueUrl(queueUrl)
                    .attributes(attrMap)
                    .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
 GetQueueAttributesRequest.builder()
```

```
                .queueUrl(queueUrl)
                .attributeNames(atts)
                .build();

        GetQueueAttributesResponse response =
 sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName, Boolean
 selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                        .queueName(queueName)
                        .attributes(attrs)
                        .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

 .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                        .queueName(queueName)
                        .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

 .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Serverless examples

Invoke a Lambda function from an Amazon SQS trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving messages from an SQS queue. The function retrieves the messages from the event parameter and logs the content of each message.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an SQS event with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }

    }
}
```

Reporting batch item failures for Lambda functions with an Amazon SQS trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from an SQS queue. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Reporting SQS batch item failures with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
 SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

         List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<SQSBatchResponse.BatchItemFailure>();
         String messageId = "";
         for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
             try {
                 //process your message
                 messageId = message.getMessageId();
             } catch (Exception e) {
                 //Add failed message identifier to the batchItemFailures list
                 batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(messageId));
```

```
                }
            }
            return new SQSBatchResponse(batchItemFailures);
        }
}
```

Step Functions examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Step Functions.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Step Functions

The following code examples show how to get started using Step Functions.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sfn#code-examples).

Java version of Hello.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
```

```
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
 machine.name());
                System.out.println("The ARN value is : " +
 machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListStateMachines in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

`CreateActivity`

The following code example shows how to use `CreateActivity`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createActivity(SfnClient sfnClient, String activityName) {
        try {
            CreateActivityRequest activityRequest = CreateActivityRequest.builder()
                    .name(activityName)
                    .build();

            CreateActivityResponse response =
 sfnClient.createActivity(activityRequest);
            return response.activityArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateActivity in *AWS SDK for Java 2.x API Reference*.

CreateStateMachine

The following code example shows how to use `CreateStateMachine`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createMachine(SfnClient sfnClient, String roleARN, String
 stateMachineName, String json) {
        try {
            CreateStateMachineRequest machineRequest =
 CreateStateMachineRequest.builder()
                    .definition(json)
                    .name(stateMachineName)
                    .roleArn(roleARN)
                    .type(StateMachineType.STANDARD)
                    .build();

            CreateStateMachineResponse response =
 sfnClient.createStateMachine(machineRequest);
            return response.stateMachineArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateStateMachine in *AWS SDK for Java 2.x API Reference*.

DeleteActivity

The following code example shows how to use `DeleteActivity`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sfn#code-examples).

```
public static void deleteActivity(SfnClient sfnClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
                .activityArn(actArn)
                .build();

        sfnClient.deleteActivity(activityRequest);
        System.out.println("You have deleted " + actArn);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteActivity]() in *AWS SDK for Java 2.x API Reference*.

`DeleteStateMachine`

The following code example shows how to use `DeleteStateMachine`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository]().

```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
```

```
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
                    .stateMachineArn(stateMachineArn)
                    .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
                    .stateMachineArn(stateMachineArn)
                    .build();

            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
                System.out.println("The state machine is not deleted yet. The status
is " + response.status());
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
        System.out.println(stateMachineArn + " was successfully deleted.");
    }
```

- For API details, see [DeleteStateMachine](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeExecution`

The following code example shows how to use `DescribeExecution`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void describeExe(SfnClient sfnClient, String executionArn) {
```

```
        try {
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                    .executionArn(executionArn)
                    .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
wait for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") == 0) {
                    System.out.println("The Step Function workflow has succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or
succeeded");
                }
            }
            System.out.println("The Status is " + status);

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeExecution in *AWS SDK for Java 2.x API Reference*.

DescribeStateMachine

The following code example shows how to use `DescribeStateMachine`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/sfn#code-examples).

```
   public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
       try {
           DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
                   .stateMachineArn(stateMachineArn)
                   .build();

           DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
           System.out.println("The name of the State machine is " +
response.name());
           System.out.println("The status of the State machine is " +
response.status());
           System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
           System.out.println("The role ARN value is " + response.roleArn());

       } catch (SfnException e) {
           System.err.println(e.getMessage());
       }
   }
```

- For API details, see [DescribeStateMachine](https://docs.aws.amazon.com/goto/SdkForJavaV2/states-2016-11-23/DescribeStateMachine) in *AWS SDK for Java 2.x API Reference*.

`GetActivityTask`

The following code example shows how to use `GetActivityTask`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
        List<String> myList = new ArrayList<>();
        GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
                .activityArn(actArn)
                .build();

        GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
        myList.add(response.taskToken());
        myList.add(response.input());
        return myList;
    }

    /// <summary>
    /// Stop execution of a Step Functions workflow.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of
    /// the Step Functions execution to stop.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StopExecution(string executionArn)
    {
        var response =
            await _amazonStepFunctions.StopExecutionAsync(new StopExecutionRequest
{ ExecutionArn = executionArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- For API details, see GetActivityTask in *AWS SDK for Java 2.x API Reference*.

ListActivities

The following code example shows how to use `ListActivities`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListActivitiesRequest;
import software.amazon.awssdk.services.sfn.model.ListActivitiesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.ActivityListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListActivities {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listAllActivites(sfnClient);
        sfnClient.close();
    }

    public static void listAllActivites(SfnClient sfnClient) {
        try {
            ListActivitiesRequest activitiesRequest =
 ListActivitiesRequest.builder()
```

```
                    .maxResults(10)
                    .build();

            ListActivitiesResponse response =
 sfnClient.listActivities(activitiesRequest);
            List<ActivityListItem> items = response.activities();
            for (ActivityListItem item : items) {
                System.out.println("The activity ARN is " + item.activityArn());
                System.out.println("The activity name is " + item.name());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListActivities in *AWS SDK for Java 2.x API Reference*.

`ListExecutions`

The following code example shows how to use `ListExecutions`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getExeHistory(SfnClient sfnClient, String exeARN) {
        try {
            GetExecutionHistoryRequest historyRequest =
 GetExecutionHistoryRequest.builder()
                    .executionArn(exeARN)
                    .maxResults(10)
                    .build();
```

```
            GetExecutionHistoryResponse historyResponse =
 sfnClient.getExecutionHistory(historyRequest);
            List<HistoryEvent> events = historyResponse.events();
            for (HistoryEvent event : events) {
                System.out.println("The event type is " + event.type().toString());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListExecutions in *AWS SDK for Java 2.x API Reference*.

ListStateMachines

The following code example shows how to use `ListStateMachines`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
 machine.name());
                System.out.println("The ARN value is : " +
 machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListStateMachines in *AWS SDK for Java 2.x API Reference*.

SendTaskSuccess

The following code example shows how to use `SendTaskSuccess`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void sendTaskSuccess(SfnClient sfnClient, String token, String
 json) {
        try {
            SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
                    .taskToken(token)
                    .output(json)
                    .build();

            sfnClient.sendTaskSuccess(successRequest);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see SendTaskSuccess in *AWS SDK for Java 2.x API Reference*.

`StartExecution`

The following code example shows how to use `StartExecution`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
 String jsonEx) {
        UUID uuid = UUID.randomUUID();
        String uuidValue = uuid.toString();
        try {
            StartExecutionRequest executionRequest = StartExecutionRequest.builder()
                    .input(jsonEx)
                    .stateMachineArn(stateMachineArn)
                    .name(uuidValue)
                    .build();

            StartExecutionResponse response =
 sfnClient.startExecution(executionRequest);
            return response.executionArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see StartExecution in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with state machines

The following code example shows how to:

- Create an activity.
- Create a state machine from an Amazon States Language definition that contains the previously created activity as a step.
- Run the state machine and respond to the activity with user input.
- Get the final status and output after the run completes, then clean up resources.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 *
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an activity.
 * 2. Creates a state machine.
 * 3. Describes the state machine.
 * 4. Starts execution of the state machine and interacts with it.
 * 5. Describes the execution.
 * 6. Delete the activity.
 * 7. Deletes the state machine.
 */
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws Exception {
        final String usage = """

                Usage:
                    <roleARN> <activityName> <stateMachineName>
```

```
                Where:
                    roleName - The name of the IAM role to create for this state
 machine.
                    activityName - The name of an activity to create.
                    stateMachineName - The name of the state machine to create.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String activityName = args[1];
        String stateMachineName = args[2];
        String polJSON = "{\n" +
                "    \"Version\": \"2012-10-17\",\n" +
                "    \"Statement\": [\n" +
                "        {\n" +
                "            \"Sid\": \"\",\n" +
                "            \"Effect\": \"Allow\",\n" +
                "            \"Principal\": {\n" +
                "                \"Service\": \"states.amazonaws.com\"\n" +
                "            },\n" +
                "            \"Action\": \"sts:AssumeRole\"\n" +
                "        }\n" +
                "    ]\n" +
                "}";

        Scanner sc = new Scanner(System.in);
        boolean action = false;

        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(regionGl)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Step Functions example scenario.");
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an activity.");
        String activityArn = createActivity(sfnClient, activityName);
        System.out.println("The ARN of the activity is " + activityArn);
        System.out.println(DASHES);

        // Get JSON to use for the state machine and place the activityArn value
 into
        // it.
        InputStream input = StepFunctionsScenario.class.getClassLoader()
                .getResourceAsStream("chat_sfn_state_machine.json");
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonNode = mapper.readValue(input, JsonNode.class);
        String jsonString = mapper.writeValueAsString(jsonNode);

        // Modify the Resource node.
        ObjectMapper objectMapper = new ObjectMapper();
        JsonNode root = objectMapper.readTree(jsonString);
        ((ObjectNode) root.path("States").path("GetInput")).put("Resource",
 activityArn);

        // Convert the modified Java object back to a JSON string.
        String stateDefinition = objectMapper.writeValueAsString(root);
        System.out.println(stateDefinition);

        System.out.println(DASHES);
        System.out.println("2. Create a state machine.");
        String roleARN = createIAMRole(iam, roleName, polJSON);
        String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
 stateDefinition);
        System.out.println("The ARN of the state machine is " + stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe the state machine.");
        describeStateMachine(sfnClient, stateMachineArn);
        System.out.println("What should ChatSFN call you?");
        String userName = sc.nextLine();
        System.out.println("Hello " + userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        // The JSON to pass to the StartExecution call.
        String executionJson = "{ \"name\" : \"" + userName + "\" }";
        System.out.println(executionJson);
        System.out.println("4. Start execution of the state machine and interact
 with it.");
        String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
        System.out.println("The ARN of the state machine execution is " + runArn);
        List<String> myList;
        while (!action) {
            myList = getActivityTask(sfnClient, activityArn);
            System.out.println("ChatSFN: " + myList.get(1));
            System.out.println(userName + " please specify a value.");
            String myAction = sc.nextLine();
            if (myAction.compareTo("done") == 0)
                action = true;

            System.out.println("You have selected " + myAction);
            String taskJson = "{ \"action\" : \"" + myAction + "\" }";
            System.out.println(taskJson);
            sendTaskSuccess(sfnClient, myList.get(0), taskJson);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the execution.");
        describeExe(sfnClient, runArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Delete the activity.");
        deleteActivity(sfnClient, activityArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the state machines.");
        deleteMachine(sfnClient, stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Step Functions example scenario is complete.");
        System.out.println(DASHES);
    }
```

```
    public static String createIAMRole(IamClient iam, String rolename, String
 polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(polJSON)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void describeExe(SfnClient sfnClient, String executionArn) {
        try {
            DescribeExecutionRequest executionRequest =
 DescribeExecutionRequest.builder()
                    .executionArn(executionArn)
                    .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
 sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
 wait for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") == 0) {
                    System.out.println("The Step Function workflow has succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or
 succeeded");
                }
            }
```

```
            System.out.println("The Status is " + status);

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void sendTaskSuccess(SfnClient sfnClient, String token, String
 json) {
        try {
            SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
                    .taskToken(token)
                    .output(json)
                    .build();

            sfnClient.sendTaskSuccess(successRequest);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
        List<String> myList = new ArrayList<>();
        GetActivityTaskRequest getActivityTaskRequest =
 GetActivityTaskRequest.builder()
                .activityArn(actArn)
                .build();

        GetActivityTaskResponse response =
 sfnClient.getActivityTask(getActivityTaskRequest);
        myList.add(response.taskToken());
        myList.add(response.input());
        return myList;
    }

    public static void deleteActivity(SfnClient sfnClient, String actArn) {
        try {
            DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
                    .activityArn(actArn)
                    .build();
```

```
            sfnClient.deleteActivity(activityRequest);
            System.out.println("You have deleted " + actArn);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeStateMachine(SfnClient sfnClient, String
 stateMachineArn) {
        try {
            DescribeStateMachineRequest stateMachineRequest =
 DescribeStateMachineRequest.builder()
                    .stateMachineArn(stateMachineArn)
                    .build();

            DescribeStateMachineResponse response =
 sfnClient.describeStateMachine(stateMachineRequest);
            System.out.println("The name of the State machine is " +
 response.name());
            System.out.println("The status of the State machine is " +
 response.status());
            System.out.println("The ARN value of the State machine is " +
 response.stateMachineArn());
            System.out.println("The role ARN value is " + response.roleArn());

        } catch (SfnException e) {
            System.err.println(e.getMessage());
        }
    }

    public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
 DeleteStateMachineRequest.builder()
                    .stateMachineArn(stateMachineArn)
                    .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
 DescribeStateMachineRequest.builder()
                    .stateMachineArn(stateMachineArn)
                    .build();
```

```
            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
                System.out.println("The state machine is not deleted yet. The status
is " + response.status());
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
        System.out.println(stateMachineArn + " was successfully deleted.");
    }

    public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
        UUID uuid = UUID.randomUUID();
        String uuidValue = uuid.toString();
        try {
            StartExecutionRequest executionRequest = StartExecutionRequest.builder()
                    .input(jsonEx)
                    .stateMachineArn(stateMachineArn)
                    .name(uuidValue)
                    .build();

            StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
            return response.executionArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
        try {
            CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
                    .definition(json)
                    .name(stateMachineName)
```

```
                    .roleArn(roleARN)
                    .type(StateMachineType.STANDARD)
                    .build();

            CreateStateMachineResponse response =
 sfnClient.createStateMachine(machineRequest);
            return response.stateMachineArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createActivity(SfnClient sfnClient, String activityName) {
        try {
            CreateActivityRequest activityRequest = CreateActivityRequest.builder()
                    .name(activityName)
                    .build();

            CreateActivityResponse response =
 sfnClient.createActivity(activityRequest);
            return response.activityArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine

- GetActivityTask
- ListActivities
- ListStateMachines
- SendTaskSuccess
- StartExecution
- StopExecution

AWS STS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS STS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

Actions

`AssumeRole`

The following code example shows how to use `AssumeRole`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 * "Version": "2012-10-17",
 * "Statement": [
 * {
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to assume
(for example, rn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
(for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn, String
roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
```

```
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " + exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see AssumeRole in *AWS SDK for Java 2.x API Reference*.

AWS Support examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Support.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello AWS Support

The following code examples show how to get started using AWS Support.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SupportException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 *
 * NOTE: To see multiple operations, see SupportScenario.
 */

public class HelloSupport {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
```

```
        SupportClient supportClient = SupportClient.builder()
                .region(region)
                .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
 DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
 supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());

                // Display the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                }
                index++;
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DescribeServices in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

AddAttachmentsToSet

The following code example shows how to use `AddAttachmentsToSet`.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
       try {
           File myFile = new File(fileAttachment);
           InputStream sourceStream = new FileInputStream(myFile);
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           Attachment attachment = Attachment.builder()
                   .fileName(myFile.getName())
                   .data(sourceBytes)
                   .build();

           AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                   .attachments(attachment)
                   .build();

           AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
           return response.attachmentSetId();
```

```
        } catch (SupportException | FileNotFoundException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see [AddAttachmentsToSet](#) in *AWS SDK for Java 2.x API Reference*.

AddCommunicationToCase

The following code example shows how to use AddCommunicationToCase.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void addAttachSupportCase(SupportClient supportClient, String
 caseId, String attachmentSetId) {
        try {
            AddCommunicationToCaseRequest caseRequest =
 AddCommunicationToCaseRequest.builder()
                    .caseId(caseId)
                    .attachmentSetId(attachmentSetId)
                    .communicationBody("Please refer to attachment for details.")
                    .build();

            AddCommunicationToCaseResponse response =
 supportClient.addCommunicationToCase(caseRequest);
            if (response.result())
                System.out.println("You have successfully added a communication to
 an AWS Support case");
            else
                System.out.println("There was an error adding the communication to
 an AWS Support case");
```

```
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see AddCommunicationToCase in *AWS SDK for Java 2.x API Reference*.

CreateCase

The following code example shows how to use `CreateCase`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createSupportCase(SupportClient supportClient, List<String>
 sevCatList, String sevLevel) {
        try {
            String serviceCode = sevCatList.get(0);
            String caseCat = sevCatList.get(1);
            CreateCaseRequest caseRequest = CreateCaseRequest.builder()
                    .categoryCode(caseCat.toLowerCase())
                    .serviceCode(serviceCode.toLowerCase())
                    .severityCode(sevLevel.toLowerCase())
                    .communicationBody("Test issue with " +
 serviceCode.toLowerCase())
                    .subject("Test case, please ignore")
                    .language("en")
                    .issueType("technical")
                    .build();

            CreateCaseResponse response = supportClient.createCase(caseRequest);
            return response.caseId();
```

```
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see [CreateCase](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeAttachment`

The following code example shows how to use `DescribeAttachment`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void describeAttachment(SupportClient supportClient, String
 attachId) {
        try {
            DescribeAttachmentRequest attachmentRequest =
 DescribeAttachmentRequest.builder()
                    .attachmentId(attachId)
                    .build();

            DescribeAttachmentResponse response =
 supportClient.describeAttachment(attachmentRequest);
            System.out.println("The name of the file is " +
 response.attachment().fileName());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeAttachment in *AWS SDK for Java 2.x API Reference*.

DescribeCases

The following code example shows how to use `DescribeCases`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getOpenCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
 DescribeCasesRequest.builder()
                    .maxResults(20)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .build();

            DescribeCasesResponse response =
 supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                System.out.println("The case status is " + sinCase.status());
                System.out.println("The case Id is " + sinCase.caseId());
                System.out.println("The case subject is " + sinCase.subject());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeCases in *AWS SDK for Java 2.x API Reference*.

DescribeCommunications

The following code example shows how to use DescribeCommunications.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String listCommunications(SupportClient supportClient, String caseId) {
        try {
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest = DescribeCommunicationsRequest.builder()
                    .caseId(caseId)
                    .maxResults(10)
                    .build();

            DescribeCommunicationsResponse response = supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm : communications) {
                System.out.println("the body is: " + comm.body());

                // Get the attachment id value.
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
                }
            }
            return attachId;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
        return "";
    }
```

- For API details, see [DescribeCommunications](#) in *AWS SDK for Java 2.x API Reference*.

`DescribeServices`

The following code example shows how to use `DescribeServices`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
 DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
 supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());
```

```
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();

                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
                index++;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return null;
    }
```

- For API details, see DescribeServices in *AWS SDK for Java 2.x API Reference*.

DescribeSeverityLevels

The following code example shows how to use `DescribeSeverityLevels`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String displaySevLevels(SupportClient supportClient) {
        try {
```

```
            DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                    .language("en")
                    .build();

            DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
            List<SeverityLevel> severityLevels = response.severityLevels();
            String levelName = null;
            for (SeverityLevel sevLevel : severityLevels) {
                System.out.println("The severity level name is: " +
sevLevel.name());
                if (sevLevel.name().compareTo("High") == 0)
                    levelName = sevLevel.name();
            }
            return levelName;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see DescribeSeverityLevels in *AWS SDK for Java 2.x API Reference*.

ResolveCase

The following code example shows how to use `ResolveCase`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
        try {
```

```
            ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                    .caseId(caseId)
                    .build();

            ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
            System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see ResolveCase in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with cases

The following code example shows how to:

- Get and display available services and severity levels for cases.
- Create a support case using a selected service, category, and severity level.
- Get and display a list of open cases for the current day.
- Add an attachment set and a communication to the new case.
- Describe the new attachment and communication for the case.
- Resolve the case.
- Get and display a list of resolved cases for the current day.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run various AWS Support operations.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SeverityLevel;
import software.amazon.awssdk.services.support.model.SupportException;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
 * 3. Creates a support case by using the selected service, category, and
 * severity level.
 * 4. Gets a list of open cases for the current day.
 * 5. Creates an attachment set with a generated file.
 * 6. Adds a communication with the attachment to the support case.
 * 7. Lists the communications of the support case.
 * 8. Describes the attachment set included with the communication.
 * 9. Resolves the support case.
 * 10. Gets a list of resolved cases for the current day.
 */
public class SupportScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <fileAttachment>Where:
                    fileAttachment - The file can be a simple saved .txt file to use
 as an email attachment.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String fileAttachment = args[0];
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
```

```
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("***** Welcome to the AWS Support case example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Get and display available services.");
        List<String> sevCatList = displayServices(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get and display Support severity levels.");
        String sevLevel = displaySevLevels(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a support case using the selected service,
category, and severity level.");
        String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
        if (caseId.compareTo("") == 0) {
            System.out.println("A support case was not successfully created!");
            System.exit(1);
        } else
            System.out.println("Support case " + caseId + " was successfully
created!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get open support cases.");
        getOpenCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to add
to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("6. Add communication with the attachment to the support
 case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
 communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
 DescribeCasesRequest.builder()
                    .maxResults(30)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
```

```
                    .includeResolvedCases(true)
                    .build();

            DescribeCasesResponse response =
 supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                if (sinCase.status().compareTo("resolved") == 0)
                    System.out.println("The case status is " + sinCase.status());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void resolveSupportCase(SupportClient supportClient, String
 caseId) {
        try {
            ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                    .caseId(caseId)
                    .build();

            ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
            System.out.println("The status of case " + caseId + " is " +
 response.finalCaseStatus());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeAttachment(SupportClient supportClient, String
 attachId) {
        try {
            DescribeAttachmentRequest attachmentRequest =
 DescribeAttachmentRequest.builder()
                    .attachmentId(attachId)
                    .build();

            DescribeAttachmentResponse response =
 supportClient.describeAttachment(attachmentRequest);
```

```
            System.out.println("The name of the file is " +
response.attachment().fileName());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String listCommunications(SupportClient supportClient, String
caseId) {
        try {
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                    .caseId(caseId)
                    .maxResults(10)
                    .build();

            DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm : communications) {
                System.out.println("the body is: " + comm.body());

                // Get the attachment id value.
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
                }
            }
            return attachId;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
        try {
```

```
            AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
                    .caseId(caseId)
                    .attachmentSetId(attachmentSetId)
                    .communicationBody("Please refer to attachment for details.")
                    .build();

            AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
            if (response.result())
                System.out.println("You have successfully added a communication to
an AWS Support case");
            else
                System.out.println("There was an error adding the communication to
an AWS Support case");

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
        try {
            File myFile = new File(fileAttachment);
            InputStream sourceStream = new FileInputStream(myFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            Attachment attachment = Attachment.builder()
                    .fileName(myFile.getName())
                    .data(sourceBytes)
                    .build();

            AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                    .attachments(attachment)
                    .build();

            AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
            return response.attachmentSetId();

        } catch (SupportException | FileNotFoundException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static void getOpenCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
 DescribeCasesRequest.builder()
                    .maxResults(20)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .build();

            DescribeCasesResponse response =
 supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                System.out.println("The case status is " + sinCase.status());
                System.out.println("The case Id is " + sinCase.caseId());
                System.out.println("The case subject is " + sinCase.subject());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String createSupportCase(SupportClient supportClient, List<String>
 sevCatList, String sevLevel) {
        try {
            String serviceCode = sevCatList.get(0);
            String caseCat = sevCatList.get(1);
            CreateCaseRequest caseRequest = CreateCaseRequest.builder()
                    .categoryCode(caseCat.toLowerCase())
                    .serviceCode(serviceCode.toLowerCase())
                    .severityCode(sevLevel.toLowerCase())
```

```
                    .communicationBody("Test issue with " +
serviceCode.toLowerCase())
                    .subject("Test case, please ignore")
                    .language("en")
                    .issueType("technical")
                    .build();

            CreateCaseResponse response = supportClient.createCase(caseRequest);
            return response.caseId();

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static String displaySevLevels(SupportClient supportClient) {
        try {
            DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                    .language("en")
                    .build();

            DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
            List<SeverityLevel> severityLevels = response.severityLevels();
            String levelName = null;
            for (SeverityLevel sevLevel : severityLevels) {
                System.out.println("The severity level name is: " +
sevLevel.name());
                if (sevLevel.name().compareTo("High") == 0)
                    levelName = sevLevel.name();
            }
            return levelName;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Return a List that contains a Service name and Category name.
```

```
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();

                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
                index++;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return null;
```

```
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Systems Manager examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Systems Manager.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Systems Manager

The following code examples show how to get started using Systems Manager.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ssm#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DocumentFilter;
import software.amazon.awssdk.services.ssm.model.ListDocumentsRequest;
import software.amazon.awssdk.services.ssm.model.ListDocumentsResponse;

public class HelloSSM {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <awsAccount>

                Where:
                    awsAccount - Your AWS Account number.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String awsAccount = args[0] ;
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        listDocuments(ssmClient, awsAccount);
    }

    /*
```

```
    This code automatically fetches the next set of results using the `nextToken`
 and
    stops once the desired maxResults (20 in this case) have been reached.
    */
    public static void listDocuments(SsmClient ssmClient, String awsAccount) {
        String nextToken = null;
        int totalDocumentsReturned = 0;
        int maxResults = 20;
        do {
            ListDocumentsRequest request = ListDocumentsRequest.builder()
                .documentFilterList(
                    DocumentFilter.builder()
                        .key("Owner")
                        .value(awsAccount)
                        .build()
                    )
                .maxResults(maxResults)
                .nextToken(nextToken)
                .build();

            ListDocumentsResponse response = ssmClient.listDocuments(request);
            response.documentIdentifiers().forEach(identifier ->
 System.out.println("Document Name: " + identifier.name()));
            nextToken = response.nextToken();
            totalDocumentsReturned += response.documentIdentifiers().size();
        } while (nextToken != null && totalDocumentsReturned < maxResults);
    }
}
```

- For API details, see listThings in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions
- Scenarios

Actions

`CreateDocument`

The following code example shows how to use `CreateDocument`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ssm#code-examples).

```
    // Create an AWS SSM document to use in this scenario.
    public static void createSSMDoc(SsmClient ssmClient, String docName) {
        // Create JSON for the content
        String jsonData = """
            {
                "schemaVersion": "2.2",
                "description": "Run a simple shell command",
                "mainSteps": [
                    {
                        "action": "aws:runShellScript",
                        "name": "runEchoCommand",
                        "inputs": {
                          "runCommand": [
                            "echo 'Hello, world!'"
                          ]
                        }
                      }
                    ]
                }
            """;

        try {
            CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
                .documentType(DocumentType.COMMAND)
                .build();

            // Create the document.
            CreateDocumentResponse response = ssmClient.createDocument(request);
            System.out.println("The status of the document is " +
 response.documentDescription().status());

        } catch (DocumentAlreadyExistsException e) {
```

```
            System.err.println("The document already exists. Moving on." );
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see [CreateDocument](#) in *AWS SDK for Java 2.x API Reference*.

`CreateMaintenanceWindow`

The following code example shows how to use `CreateMaintenanceWindow`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static String createMaintenanceWindow(SsmClient ssmClient, String
 winName) {
        CreateMaintenanceWindowRequest request =
 CreateMaintenanceWindowRequest.builder()
            .name(winName)
            .description("This is my maintenance window")
            .allowUnassociatedTargets(true)
            .duration(2)
            .cutoff(1)
            .schedule("cron(0 10 ? * MON-FRI *)")
            .build();

        try {
            CreateMaintenanceWindowResponse response =
 ssmClient.createMaintenanceWindow(request);
            String maintenanceWindowId = response.windowId();
            System.out.println("The maintenance window id is " +
 maintenanceWindowId);
            return maintenanceWindowId;
```

```
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The maintenance window already exists. Moving on.");
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
            .key("name")
            .values(winName)
            .build();

        DescribeMaintenanceWindowsRequest winRequest =
 DescribeMaintenanceWindowsRequest.builder()
            .filters(filter)
            .build();

        String windowId = "";
        DescribeMaintenanceWindowsResponse response =
 ssmClient.describeMaintenanceWindows(winRequest);
        List<MaintenanceWindowIdentity> windows = response.windowIdentities();
        if (!windows.isEmpty()) {
            windowId = windows.get(0).windowId();
            System.out.println("Window ID: " + windowId);
        } else {
            System.out.println("Window not found.");
        }
       return windowId;
    }
```

- For API details, see CreateMaintenanceWindow in *AWS SDK for Java 2.x API Reference*.

`CreateOpsItem`

The following code example shows how to use `CreateOpsItem`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ssm#code-examples).

```
    // Create an SSM OpsItem
    public static String createSSMOpsItem(SsmClient ssmClient, String title, String
 source, String category, String severity) {
        try {
            CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the Systems Manager Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

            CreateOpsItemResponse itemResponse =
 ssmClient.createOpsItem(opsItemRequest);
            return itemResponse.opsItemId();

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
       return "";
    }
```

- For API details, see CreateOpsItem in *AWS SDK for Java 2.x API Reference*.

`DeleteDocument`

The following code example shows how to use `DeleteDocument`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Deletes an AWS Systems Manager document.
    public static void deleteDoc(SsmClient ssmClient, String documentName) {
        try {
            DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

            ssmClient.deleteDocument(documentRequest);
            System.out.println("The Systems Manager document was successfully
 deleted.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteDocument in *AWS SDK for Java 2.x API Reference*.

`DeleteMaintenanceWindow`

The following code example shows how to use `DeleteMaintenanceWindow`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteMaintenanceWindow(SsmClient ssmClient, String winId) {
        try {
            DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
                .windowId(winId)
                .build();

            ssmClient.deleteMaintenanceWindow(windowRequest);
            System.out.println("The maintenance window was successfully deleted.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteMaintenanceWindow in *AWS SDK for Java 2.x API Reference*.

`DescribeOpsItems`

The following code example shows how to use `DescribeOpsItems`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeOpsItems(SsmClient ssmClient, String key) {
        try {
            OpsItemFilter filter = OpsItemFilter.builder()
                .key(OpsItemFilterKey.OPS_ITEM_ID)
                .values(key)
                .operator(OpsItemFilterOperator.EQUAL)
                .build();

            DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
```

```
                .opsItemFilters(filter)
                .build();

            DescribeOpsItemsResponse itemsResponse =
 ssmClient.describeOpsItems(itemsRequest);
            List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
            for (OpsItemSummary item : items) {
                System.out.println("The item title is " + item.title() +" and the
 status is "+item.status().toString());
            }

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DescribeOpsItems](https://docs.aws.amazon.com/goto/SdkForJavaV2/ssm-2014-11-06/DescribeOpsItems) in *AWS SDK for Java 2.x API Reference*.

`DescribeParameters`

The following code example shows how to use `DescribeParameters`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ssm#code-examples).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.GetParameterRequest;
import software.amazon.awssdk.services.ssm.model.GetParameterResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetParameter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String paraName = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        getParaValue(ssmClient, paraName);
        ssmClient.close();
    }

    public static void getParaValue(SsmClient ssmClient, String paraName) {
        try {
            GetParameterRequest parameterRequest = GetParameterRequest.builder()
                    .name(paraName)
                    .build();

            GetParameterResponse parameterResponse =
 ssmClient.getParameter(parameterRequest);
            System.out.println("The parameter value is " +
 parameterResponse.parameter().value());

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
}
```

- For API details, see DescribeParameters in *AWS SDK for Java 2.x API Reference*.

PutParameter

The following code example shows how to use `PutParameter`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.ParameterType;
import software.amazon.awssdk.services.ssm.model.PutParameterRequest;
import software.amazon.awssdk.services.ssm.model.SsmException;

public class PutParameter {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                    paraValue - The value of the parameter.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String paraName = args[0];
        String paraValue = args[1];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        putParaValue(ssmClient, paraName, paraValue);
        ssmClient.close();
    }

    public static void putParaValue(SsmClient ssmClient, String paraName, String
 value) {
        try {
            PutParameterRequest parameterRequest = PutParameterRequest.builder()
                    .name(paraName)
                    .type(ParameterType.STRING)
                    .value(value)
                    .build();

            ssmClient.putParameter(parameterRequest);
            System.out.println("The parameter was successfully added.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [PutParameter](#) in *AWS SDK for Java 2.x API Reference*.

SendCommand

The following code example shows how to use `SendCommand`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Sends a SSM command to a managed node.
    public static String sendSSMCommand(SsmClient ssmClient, String documentName,
 String instanceId) throws InterruptedException {
        // Before we use Document to send a command - make sure it is active.
        boolean isDocumentActive = false;
        DescribeDocumentRequest request = DescribeDocumentRequest.builder()
            .name(documentName)
            .build();

        while (!isDocumentActive) {
            DescribeDocumentResponse response = ssmClient.describeDocument(request);
            String documentStatus = response.document().statusAsString();
            if (documentStatus.equals("Active")) {
                System.out.println("The Systems Manager document is active and ready
 to use.");
                isDocumentActive = true;
            } else {
                System.out.println("The Systems Manager document is not active.
 Status: " + documentStatus);
                try {
                    // Add a delay to avoid making too many requests.
                    Thread.sleep(5000); // Wait for 5 seconds before checking again
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }

        // Create the SendCommandRequest.
        SendCommandRequest commandRequest = SendCommandRequest.builder()
            .documentName(documentName)
            .instanceIds(instanceId)
            .build();
```

```
        // Send the command.
        SendCommandResponse commandResponse = ssmClient.sendCommand(commandRequest);
        String commandId = commandResponse.command().commandId();
        System.out.println("The command Id is " + commandId);

        // Wait for the command execution to complete.
        GetCommandInvocationRequest invocationRequest =
 GetCommandInvocationRequest.builder()
            .commandId(commandId)
            .instanceId(instanceId)
            .build();

        System.out.println("Wait 5 secs");
        TimeUnit.SECONDS.sleep(5);

        // Retrieve the command execution details.
        GetCommandInvocationResponse commandInvocationResponse =
 ssmClient.getCommandInvocation(invocationRequest);

        // Check the status of the command execution.
        CommandInvocationStatus status = commandInvocationResponse.status();
        if (status == CommandInvocationStatus.SUCCESS) {
            System.out.println("Command execution successful.");
        } else {
            System.out.println("Command execution failed. Status: " + status);
        }
        return commandId;
    }
```

- For API details, see SendCommand in *AWS SDK for Java 2.x API Reference*.

UpdateMaintenanceWindow

The following code example shows how to use `UpdateMaintenanceWindow`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Update the maintenance window schedule
    public static void updateSSMMaintenanceWindow(SsmClient ssmClient, String id,
 String name) {
        try {
            UpdateMaintenanceWindowRequest updateRequest =
 UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();

            ssmClient.updateMaintenanceWindow(updateRequest);
            System.out.println("The Systems Manager maintenance window was
 successfully updated.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see UpdateMaintenanceWindow in *AWS SDK for Java 2.x API Reference*.

`UpdateOpsItem`

The following code example shows how to use `UpdateOpsItem`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void resolveOpsItem(SsmClient ssmClient, String opsID) {
        try {
```

```
            UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

            ssmClient.updateOpsItem(opsItemRequest);

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see UpdateOpsItem in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with Systems Manager

The following code example shows how to work with Systems Manager maintenance windows, documents, and OpsItems.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.CommandInvocation;
import software.amazon.awssdk.services.ssm.model.CommandInvocationStatus;
import software.amazon.awssdk.services.ssm.model.CreateDocumentRequest;
import software.amazon.awssdk.services.ssm.model.CreateDocumentResponse;
import software.amazon.awssdk.services.ssm.model.CreateMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.CreateMaintenanceWindowResponse;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemResponse;
```

```
import software.amazon.awssdk.services.ssm.model.DeleteDocumentRequest;
import software.amazon.awssdk.services.ssm.model.DeleteMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.DeleteOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.DescribeDocumentRequest;
import software.amazon.awssdk.services.ssm.model.DescribeDocumentResponse;
import software.amazon.awssdk.services.ssm.model.DescribeMaintenanceWindowsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeMaintenanceWindowsResponse;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsResponse;
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.DocumentType;
import software.amazon.awssdk.services.ssm.model.GetCommandInvocationRequest;
import software.amazon.awssdk.services.ssm.model.GetCommandInvocationResponse;
import software.amazon.awssdk.services.ssm.model.GetOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.GetOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.ListCommandInvocationsRequest;
import software.amazon.awssdk.services.ssm.model.ListCommandInvocationsResponse;
import software.amazon.awssdk.services.ssm.model.MaintenanceWindowFilter;
import software.amazon.awssdk.services.ssm.model.MaintenanceWindowIdentity;
import software.amazon.awssdk.services.ssm.model.OpsItemDataValue;
import software.amazon.awssdk.services.ssm.model.OpsItemFilter;
import software.amazon.awssdk.services.ssm.model.OpsItemFilterKey;
import software.amazon.awssdk.services.ssm.model.OpsItemFilterOperator;
import software.amazon.awssdk.services.ssm.model.OpsItemStatus;
import software.amazon.awssdk.services.ssm.model.OpsItemSummary;
import software.amazon.awssdk.services.ssm.model.SendCommandRequest;
import software.amazon.awssdk.services.ssm.model.SendCommandResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;
import software.amazon.awssdk.services.ssm.model.UpdateMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.UpdateOpsItemRequest;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html
 *
 *
 * This Java program performs these tasks:
 * 1. Creates an AWS Systems Manager maintenance window with a default name or a
 user-provided name.
 * 2. Modifies the maintenance window schedule.
 * 3. Creates a Systems Manager document with a default name or a user-provided
 name.
 * 4. Sends a command to a specified EC2 instance using the created Systems Manager
 document and displays the time when the command was invoked.
 * 5. Creates a Systems Manager OpsItem with a predefined title, source, category,
 and severity.
 * 6. Updates and resolves the created OpsItem.
 * 7. Deletes the Systems Manager maintenance window, OpsItem, and document.
 */

public class SSMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {
        String usage = """
            Usage:
              <instanceId> <title> <source> <category> <severity>

            Where:
                instanceId - The Amazon EC2 Linux/UNIX instance Id that AWS Systems
 Manager uses (ie, i-0149338494ed95f06).
                title - The title of the parameter (default is Disk Space Alert).
                source - The source of the parameter (default is EC2).
                category - The category of the parameter. Valid values are
 'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
 Performance).
                severity - The severity of the parameter. Severity should be a
 number from 1 to 4 (default is 2).
        """;

       if (args.length != 1) {
           System.out.println(usage);
           System.exit(1);
       }

        Scanner scanner = new Scanner(System.in);
        String documentName;
        String windowName;
```

```
        String instanceId = args[0];
        String title = "Disk Space Alert" ;
        String source = "EC2" ;
        String category = "Performance" ;
        String severity = "2" ;

        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("""
            Welcome to the AWS Systems Manager SDK Getting Started scenario.
            This program demonstrates how to interact with Systems Manager using the
 AWS SDK for Java (v2).
            Systems Manager is the operations hub for your AWS applications and
 resources and a secure end-to-end management solution.
            The program's primary functions include creating a maintenance window,
 creating a document, sending a command to a document,
            listing documents, listing commands, creating an OpsItem, modifying an
 OpsItem, and deleting Systems Manager resources.
            Upon completion of the program, all AWS resources are cleaned up.
            Let's get started...
            Please hit Enter
            """);
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println("Create a Systems Manager maintenance window.");
        System.out.println("Please enter the maintenance window name (default is
 ssm-maintenance-window):");
        String win = scanner.nextLine();
        windowName = win.isEmpty() ? "ssm-maintenance-window" : win;
        String winId = createMaintenanceWindow(ssmClient, windowName);
        System.out.println(DASHES);

        System.out.println("Modify the maintenance window by changing the
 schedule");
        System.out.println("Please hit Enter");
        scanner.nextLine();
        updateSSMMaintenanceWindow(ssmClient, winId, windowName);
        System.out.println(DASHES);
```

```
        System.out.println("Create a document that defines the actions that Systems
 Manager performs on your EC2 instance.");
        System.out.println("Please enter the document name (default is
 ssmdocument):");
        String doc = scanner.nextLine();
        documentName = doc.isEmpty() ? "ssmdocument" : doc;
        createSSMDoc(ssmClient, documentName);

        System.out.println("Now we are going to run a command on an EC2 instance
 that echoes 'Hello, world!'");
        System.out.println("Please hit Enter");
        scanner.nextLine();
        String commandId = sendSSMCommand(ssmClient, documentName, instanceId);
        System.out.println(DASHES);

        System.out.println("Lets get the time when the specific command was sent to
 the specific managed node");
        System.out.println("Please hit Enter");
        scanner.nextLine();
        displayCommands(ssmClient, commandId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
             Now we will create a  Systems Manager OpsItem.
             An OpsItem is a feature provided by the Systems Manager service.
             It is a type of operational data item that allows you to manage and
 track various operational issues,
             events, or tasks within your AWS environment.

             You can create OpsItems to track and manage operational issues as they
 arise.
             For example, you could create an OpsItem whenever your application
 detects a critical error
             or an anomaly in your infrastructure.
            """);

        System.out.println("Please hit Enter");
        scanner.nextLine();
        String opsItemId = createSSMOpsItem(ssmClient, title, source, category,
 severity);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("Now we will update  the OpsItem "+opsItemId);
        System.out.println("Please hit Enter");
        scanner.nextLine();
        String description = "An update to "+opsItemId ;
        updateOpsItem(ssmClient, opsItemId, title, description);
        System.out.println("Now we will get the status of the OpsItem "+opsItemId);
        System.out.println("Please hit Enter");
        scanner.nextLine();
        describeOpsItems(ssmClient, opsItemId);
        System.out.println("Now we will resolve the OpsItem "+opsItemId);
        System.out.println("Please hit Enter");
        scanner.nextLine();
        resolveOpsItem(ssmClient, opsItemId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to delete the Systems Manager resources?
 (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the resources.");
            System.out.print("Press Enter to continue...");
            scanner.nextLine();
            deleteOpsItem(ssmClient, opsItemId);
            deleteMaintenanceWindow(ssmClient, winId);
            deleteDoc(ssmClient, documentName);
        } else {
            System.out.println("The Systems Manager resources will not be deleted");
        }
        System.out.println(DASHES);

        System.out.println("This concludes the Systems Manager SDK Getting Started
 scenario.");
        System.out.println(DASHES);
    }

    // Displays the date and time when the specific command was invoked.
    public static void displayCommands(SsmClient ssmClient, String commandId) {
        try {
            ListCommandInvocationsRequest commandInvocationsRequest =
 ListCommandInvocationsRequest.builder()
                .commandId(commandId)
                .build();
```

```
            ListCommandInvocationsResponse response =
ssmClient.listCommandInvocations(commandInvocationsRequest);
            List<CommandInvocation> commandList = response.commandInvocations();
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
            for (CommandInvocation invocation : commandList) {
                System.out.println("The time of the command invocation is " +
formatter.format(invocation.requestedDateTime()));
            }

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Create an SSM OpsItem
    public static String createSSMOpsItem(SsmClient ssmClient, String title, String
source, String category, String severity) {
        try {
            CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the Systems Manager Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

            CreateOpsItemResponse itemResponse =
ssmClient.createOpsItem(opsItemRequest);
            return itemResponse.opsItemId();

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
       return "";
    }

    // Update the AWS SSM OpsItem.
    public static void updateOpsItem(SsmClient ssmClient, String opsItemId, String
title, String description) {
        Map<String, OpsItemDataValue> operationalData = new HashMap<>();
```

```
        operationalData.put("key1",
OpsItemDataValue.builder().value("value1").build());
        operationalData.put("key2",
OpsItemDataValue.builder().value("value2").build());

        try {
            UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
                .opsItemId(opsItemId)
                .title(title)
                .operationalData(operationalData)
                .status(getOpsItem(ssmClient, opsItemId))
                .description(description)
                .build();

            ssmClient.updateOpsItem(request);

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void resolveOpsItem(SsmClient ssmClient, String opsID) {
        try {
            UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

            ssmClient.updateOpsItem(opsItemRequest);

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Gets a specific OpsItem.
    private static OpsItemStatus getOpsItem(SsmClient ssmClient, String opsItemId) {
        GetOpsItemRequest itemRequest = GetOpsItemRequest.builder()
            .opsItemId(opsItemId)
            .build();

        try {
```

```
            GetOpsItemResponse response = ssmClient.getOpsItem(itemRequest);
            return response.opsItem().status();

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return null;
    }

    // Sends a SSM command to a managed node.
    public static String sendSSMCommand(SsmClient ssmClient, String documentName,
String instanceId) throws InterruptedException {
        // Before we use Document to send a command - make sure it is active.
        boolean isDocumentActive = false;
        DescribeDocumentRequest request = DescribeDocumentRequest.builder()
            .name(documentName)
            .build();

        while (!isDocumentActive) {
            DescribeDocumentResponse response = ssmClient.describeDocument(request);
            String documentStatus = response.document().statusAsString();
            if (documentStatus.equals("Active")) {
                System.out.println("The Systems Manager document is active and ready
to use.");
                isDocumentActive = true;
            } else {
                System.out.println("The Systems Manager document is not active.
Status: " + documentStatus);
                try {
                    // Add a delay to avoid making too many requests.
                    Thread.sleep(5000); // Wait for 5 seconds before checking again
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }

        // Create the SendCommandRequest.
        SendCommandRequest commandRequest = SendCommandRequest.builder()
            .documentName(documentName)
            .instanceIds(instanceId)
            .build();
```

```
        // Send the command.
        SendCommandResponse commandResponse = ssmClient.sendCommand(commandRequest);
        String commandId = commandResponse.command().commandId();
        System.out.println("The command Id is " + commandId);

        // Wait for the command execution to complete.
        GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
            .commandId(commandId)
            .instanceId(instanceId)
            .build();

        System.out.println("Wait 5 secs");
        TimeUnit.SECONDS.sleep(5);

        // Retrieve the command execution details.
        GetCommandInvocationResponse commandInvocationResponse =
ssmClient.getCommandInvocation(invocationRequest);

        // Check the status of the command execution.
        CommandInvocationStatus status = commandInvocationResponse.status();
        if (status == CommandInvocationStatus.SUCCESS) {
            System.out.println("Command execution successful.");
        } else {
            System.out.println("Command execution failed. Status: " + status);
        }
        return commandId;
    }

    // Deletes an AWS Systems Manager document.
    public static void deleteDoc(SsmClient ssmClient, String documentName) {
        try {
            DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

            ssmClient.deleteDocument(documentRequest);
            System.out.println("The Systems Manager document was successfully
deleted.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }

    public static void deleteMaintenanceWindow(SsmClient ssmClient, String winId) {
        try {
            DeleteMaintenanceWindowRequest windowRequest =
 DeleteMaintenanceWindowRequest.builder()
                .windowId(winId)
                .build();

            ssmClient.deleteMaintenanceWindow(windowRequest);
            System.out.println("The maintenance window was successfully deleted.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Update the maintenance window schedule
    public static void updateSSMMaintenanceWindow(SsmClient ssmClient, String id,
 String name) {
        try {
            UpdateMaintenanceWindowRequest updateRequest =
 UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();

            ssmClient.updateMaintenanceWindow(updateRequest);
            System.out.println("The Systems Manager maintenance window was
 successfully updated.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createMaintenanceWindow(SsmClient ssmClient, String
 winName) {
```

```
        CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
            .name(winName)
            .description("This is my maintenance window")
            .allowUnassociatedTargets(true)
            .duration(2)
            .cutoff(1)
            .schedule("cron(0 10 ? * MON-FRI *)")
            .build();

        try {
            CreateMaintenanceWindowResponse response =
ssmClient.createMaintenanceWindow(request);
            String maintenanceWindowId = response.windowId();
            System.out.println("The maintenance window id is " +
maintenanceWindowId);
            return maintenanceWindowId;

        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The maintenance window already exists. Moving on.");
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
            .key("name")
            .values(winName)
            .build();

        DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
            .filters(filter)
            .build();

        String windowId = "";
        DescribeMaintenanceWindowsResponse response =
ssmClient.describeMaintenanceWindows(winRequest);
        List<MaintenanceWindowIdentity> windows = response.windowIdentities();
        if (!windows.isEmpty()) {
            windowId = windows.get(0).windowId();
            System.out.println("Window ID: " + windowId);
        } else {
            System.out.println("Window not found.");
```

```
        }
       return windowId;
    }

    // Create an AWS SSM document to use in this scenario.
    public static void createSSMDoc(SsmClient ssmClient, String docName) {
        // Create JSON for the content
        String jsonData = """
            {
                "schemaVersion": "2.2",
                "description": "Run a simple shell command",
                "mainSteps": [
                    {
                        "action": "aws:runShellScript",
                        "name": "runEchoCommand",
                        "inputs": {
                          "runCommand": [
                            "echo 'Hello, world!'"
                          ]
                        }
                      }
                    ]
                }
            """;

        try {
            CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
                .documentType(DocumentType.COMMAND)
                .build();

            // Create the document.
            CreateDocumentResponse response = ssmClient.createDocument(request);
            System.out.println("The status of the document is " +
 response.documentDescription().status());

        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The document already exists. Moving on." );
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
    public static void describeOpsItems(SsmClient ssmClient, String key) {
        try {
            OpsItemFilter filter = OpsItemFilter.builder()
                .key(OpsItemFilterKey.OPS_ITEM_ID)
                .values(key)
                .operator(OpsItemFilterOperator.EQUAL)
                .build();

            DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
                .opsItemFilters(filter)
                .build();

            DescribeOpsItemsResponse itemsResponse =
 ssmClient.describeOpsItems(itemsRequest);
            List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
            for (OpsItemSummary item : items) {
                System.out.println("The item title is " + item.title() +" and the
 status is "+item.status().toString());
            }

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteOpsItem(SsmClient ssmClient, String opsId) {
        try {
            DeleteOpsItemRequest deleteOpsItemRequest =
 DeleteOpsItemRequest.builder()
                .opsItemId(opsId)
                .build();

            ssmClient.deleteOpsItem(deleteOpsItemRequest);
            System.out.println(opsId +" Opsitem was deleted");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CommandInvocations
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - SendCommand
  - UpdateOpsItem

Amazon Textract examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Textract.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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

Actions

`AnalyzeDocument`

The following code example shows how to use `AnalyzeDocument`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentRequest;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.FeatureType;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AnalyzeDocument {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sourceDoc>\s

                Where:
```

```
                   sourceDoc - The path where the document is located (must be an
image, for example, C:/AWS/book.png).\s
               """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        analyzeDoc(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void analyzeDoc(TextractClient textractClient, String sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            List<FeatureType> featureTypes = new ArrayList<FeatureType>();
            featureTypes.add(FeatureType.FORMS);
            featureTypes.add(FeatureType.TABLES);

            AnalyzeDocumentRequest analyzeDocumentRequest =
AnalyzeDocumentRequest.builder()
                    .featureTypes(featureTypes)
                    .document(myDoc)
                    .build();

            AnalyzeDocumentResponse analyzeDocument =
textractClient.analyzeDocument(analyzeDocumentRequest);
            List<Block> docInfo = analyzeDocument.blocks();
            Iterator<Block> blockIterator = docInfo.iterator();
```

```
            while (blockIterator.hasNext()) {
                Block block = blockIterator.next();
                System.out.println("The block type is " +
 block.blockType().toString());
            }

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see AnalyzeDocument in *AWS SDK for Java 2.x API Reference*.

DetectDocumentText

The following code example shows how to use `DetectDocumentText`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Detect text from an input document.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentText {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sourceDoc>\s

                Where:
                    sourceDoc - The path where the document is located (must be an
 image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocText(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void detectDocText(TextractClient textractClient, String
 sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
            // Get the input Document object as bytes.
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            // Invoke the Detect operation.
            DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            List<Block> docInfo = textResponse.blocks();
            for (Block block : docInfo) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detect text from a document located in an Amazon S3 bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
```

```
import software.amazon.awssdk.services.textract.model.TextractException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentTextS3 {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains the
 document.\s

                    docName - The document name (must be an image, i.e., book.png).
\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocTextS3(textractClient, bucketName, docName);
        textractClient.close();
    }

    public static void detectDocTextS3(TextractClient textractClient, String
 bucketName, String docName) {
```

```
        try {
            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            // Create a Document object and reference the s3Object instance.
            Document myDoc = Document.builder()
                    .s3Object(s3Object)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            for (Block block : textResponse.blocks()) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DetectDocumentText in *AWS SDK for Java 2.x API Reference*.

StartDocumentAnalysis

The following code example shows how to use `StartDocumentAnalysis`.

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.DocumentLocation;
import software.amazon.awssdk.services.textract.model.TextractException;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.FeatureType;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartDocumentAnalysis {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains the
 document.\s
                    docName - The document name (must be an image, for example,
 book.png).\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        String jobId = startDocAnalysisS3(textractClient, bucketName, docName);
        System.out.println("Getting results for job " + jobId);
        String status = getJobResults(textractClient, jobId);
        System.out.println("The job status is " + status);
        textractClient.close();
    }

    public static String startDocAnalysisS3(TextractClient textractClient, String
 bucketName, String docName) {
        try {
            List<FeatureType> myList = new ArrayList<>();
            myList.add(FeatureType.TABLES);
            myList.add(FeatureType.FORMS);

            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            DocumentLocation location = DocumentLocation.builder()
                    .s3Object(s3Object)
                    .build();

            StartDocumentAnalysisRequest documentAnalysisRequest =
 StartDocumentAnalysisRequest.builder()
                    .documentLocation(location)
                    .featureTypes(myList)
                    .build();

            StartDocumentAnalysisResponse response =
 textractClient.startDocumentAnalysis(documentAnalysisRequest);
```

```
            // Get the job ID
            String jobId = response.jobId();
            return jobId;

        } catch (TextractException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    private static String getJobResults(TextractClient textractClient, String jobId)
{
        boolean finished = false;
        int index = 0;
        String status = "";

        try {
            while (!finished) {
                GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                        .jobId(jobId)
                        .maxResults(1000)
                        .build();

                GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
                status = response.jobStatus().toString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(index + " status is: " + status);
                    Thread.sleep(1000);
                }
                index++;
            }

            return status;

        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
```

```
        }
        return "";
    }
}
```

- For API details, see StartDocumentAnalysis in *AWS SDK for Java 2.x API Reference*.

Amazon Transcribe examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Transcribe.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

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
- Scenarios

Actions

`ListTranscriptionJobs`

The following code example shows how to use `ListTranscriptionJobs`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();

            listTranscriptionJobs(transcribeClient);
        }

        public static void listTranscriptionJobs(TranscribeClient transcribeClient)
 {
            ListTranscriptionJobsRequest listJobsRequest =
 ListTranscriptionJobsRequest.builder()
                .build();


 transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
                .flatMap(response -> response.transcriptionJobSummaries().stream())
                .forEach(jobSummary -> {
                    System.out.println("Job Name: " +
 jobSummary.transcriptionJobName());
                    System.out.println("Job Status: " +
 jobSummary.transcriptionJobStatus());
                    System.out.println("Output Location: " +
 jobSummary.outputLocationType());
                    // Add more information as needed

                    // Retrieve additional details for the job if necessary
                    GetTranscriptionJobResponse jobDetails =
 transcribeClient.getTranscriptionJob(
                        GetTranscriptionJobRequest.builder()
                            .transcriptionJobName(jobSummary.transcriptionJobName())
                            .build());

                    // Display additional details
                    System.out.println("Language Code: " +
 jobDetails.transcriptionJob().languageCode());
                    System.out.println("Media Format: " +
 jobDetails.transcriptionJob().mediaFormat());
                    // Add more details as needed

                    System.out.println("--------------");
                });
```

```
        }
    }
```

- For API details, see ListTranscriptionJobs in *AWS SDK for Java 2.x API Reference*.

`StartTranscriptionJob`

The following code example shows how to use `StartTranscriptionJob`.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {
```

```
        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
```

```
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private InputStream getStreamFromFile(String audioFileName) {
        try {
            File inputFile = new
 File(getClass().getClassLoader().getResource(audioFileName).getFile());
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
```

```
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }
```

```
        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- For API details, see StartTranscriptionJob in *AWS SDK for Java 2.x API Reference*.

Scenarios

Transcribe audio and get job data

The following code example shows how to:

- Start a transcription job with Amazon Transcribe.

- Wait for the job to complete.
- Get the URI where the transcript is stored.

For more information, see Getting started with Amazon Transcribe.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Transcribes a PCM file.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
 InterruptedException {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <file> \n\n" +
                "Where:\n" +
                "    file - the location of a PCM file to transcribe. In this
 example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
```

```
        }

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromFile(file)),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromFile(String file) {
        try {
            File inputFile = new File(file);
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US)
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
```

```
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
```

```
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
```

```
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

Transcribes streaming audio from your computer's microphone.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();
```

```
        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
```

```
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private InputStream getStreamFromFile(String audioFileName) {
        try {
            File inputFile = new
 File(getClass().getClassLoader().getResource(audioFileName).getFile());
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }
```

```
        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
 {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
 positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
 audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
```

```
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - GetTranscriptionJob
  - StartTranscriptionJob

Cross-service examples using SDK for Java 2.x

The following sample applications use the AWS SDK for Java 2.x to work across multiple AWS services.

Cross-service examples target an advanced level of experience to help you start building applications.

Examples

Build an application to submit data to a DynamoDB table

SDK for Java 2.x

Shows how to create a dynamic web application that submits data using the Amazon DynamoDB Java API and sends a text message using the Amazon Simple Notification Service Java 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

Create an Amazon Lex chatbot to engage your website visitors

SDK for Java 2.x

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Build a publish and subscription application that translates messages

SDK for Java 2.x

Shows how to use the Amazon Simple Notification Service Java API to create a web application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For complete source code and instructions on how to set up and run the example that uses the Java Async API, see the full example on GitHub.

Services used in this example

- Amazon SNS
- Amazon Translate

Create a web application that sends and retrieves messages by using Amazon SQS

SDK for Java 2.x

Shows how to use the Amazon SQS API to develop a Spring REST API that sends and retrieves messages.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Comprehend
- Amazon SQS

Create a photo asset management application that lets users manage photos using labels

SDK for Java 2.x

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Create a web application to track DynamoDB data

SDK for Java 2.x

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- DynamoDB
- Amazon SES

Create an Amazon Redshift item tracker

SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on GitHub.

Services used in this example

- Amazon Redshift
- Amazon SES

Create an Aurora Serverless work item tracker

SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on GitHub.

For complete source code and instructions on how to set up and run an example that uses the JDBC API, see the full example on GitHub.

Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Create an application that analyzes customer feedback and synthesizes audio

SDK for Java 2.x

This example application analyzes and stores customer feedback cards. Specifically, it fulfills the need of a fictitious hotel in New York City. The hotel receives feedback from guests in various languages in the form of physical comment cards. That feedback is uploaded into the app through a web client. After an image of a comment card is uploaded, the following steps occur:

- Text is extracted from the image using Amazon Textract.
- Amazon Comprehend determines the sentiment of the extracted text and its language.
- The extracted text is translated to English using Amazon Translate.
- Amazon Polly synthesizes an audio file from the extracted text.

The full app can be deployed with the AWS CDK. For source code and deployment instructions, see the project in GitHub.

Services used in this example

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Detect PPE in images with Amazon Rekognition using an AWS SDK

SDK for Java 2.x

Shows how to create an AWS Lambda function that detects images with Personal Protective Equipment.

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 Rekognition
- Amazon S3
- Amazon SES

Detect objects in images with Amazon Rekognition using an AWS SDK

SDK for Java 2.x

Shows how to use Amazon Rekognition Java 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

Detect people and objects in a video with Amazon Rekognition using an AWS SDK

SDK for Java 2.x

Shows how to use Amazon Rekognition Java API to create an app to detect faces and objects in videos 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

Monitor performance of Amazon DynamoDB using an AWS SDK

SDK for Java 2.x

This example shows how to configure a Java application to monitor the performance of DynamoDB. The application sends metric data to CloudWatch where you can monitor the performance.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- CloudWatch

- DynamoDB

Use API Gateway to invoke a Lambda function

SDK for Java 2.x

Shows how to create an AWS Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Use Step Functions to invoke Lambda functions

SDK for Java 2.x

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for Java 2.x. Each workflow step is implemented using an AWS Lambda function.

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
- Lambda
- Amazon SES

- Step Functions

Use scheduled events to invoke a Lambda function

SDK for Java 2.x

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

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
- EventBridge
- Lambda
- Amazon SNS

Security for the AWS SDK for Java

Cloud security at Amazon Web Services (AWS) is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations. Security is a shared responsibility between AWS and you. The Shared Responsibility Model describes this as Security of the Cloud and Security in the Cloud.

Security of the Cloud – AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud and providing you with services that you can use securely. Our security responsibility is the highest priority at AWS, and the effectiveness of our security is regularly tested and verified by third-party auditors as part of the AWS Compliance Programs.

Security in the Cloud – Your responsibility is determined by the AWS service you are using, and other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

Topics

- Data protection in AWS SDK for Java 2.x
- Working with TLS in the SDK for Java
- Identity and Access Management
- Compliance Validation for this AWS Product or Service
- Resilience for this AWS Product or Service
- Infrastructure Security for this AWS Product or Service

Data protection in AWS SDK for Java 2.x

The AWS shared responsibility model applies to data protection in AWS SDK for Java. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 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 confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with SDK for Java or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Working with TLS in the SDK for Java

The AWS SDK for Java uses the TLS capabilities of its underlying Java platform. In this topic, we show examples using the OpenJDK implementation used by Amazon Corretto 17.

To work with AWS services, the underlying JDK must support a minimum version of TLS 1.2, but TLS 1.3 is recommended.

Users should consult the documentation of the the Java platform they are using with the SDK to find out which TLS versions are enabled by default as well as how to enable and disable specific TLS versions.

How to check TLS version information

Using OpenJDK, the following code shows the use of SSLContext to print which TLS/SSL versions are supported.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProto
```

For example, Amazon Corretto 17 (OpenJDK) produces the following output.

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

To see the SSL handshake in action and what version of TLS is used, you can use the system property **javax.net.debug**.

For example, run a Java applications that uses TLS.

```
java app.jar -Djavax.net.debug=ssl:handshake
```

The application logs the SSL handshake similar to the following.

```
...
javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.221 EST|ClientHello.java:641|Produced
 ClientHello handshake message (
"ClientHello": {
  "client version"      : "TLSv1.2",

...
javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.295 EST|ServerHello.java:888|Consuming
 ServerHello handshake message (
"ServerHello": {
  "server version"      : "TLSv1.2",
...
```

Enforce a minimum TLS version

The SDK for Java always prefers the latest TLS version supported by the platform and service. If you wish to enforce a specific minimum TLS version, consult your Java platform's documentation.

For OpenJDK-based JVMs, you can use the system property `jdk.tls.client.protocols`.

For example, if you want SDK service clients in your application to use TLS 1.2, even though TLS 1.3 is available, provide the following system property.

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

AWS API endpoints upgrade to TLS 1.2

See this blog post for information about AWS API endpoints moving to TLS 1.2 for the minimum version.

Identity and Access Management

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How AWS services work with IAM
- Troubleshooting AWS identity and access

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS.

Service user – If you use AWS services to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in AWS, see Troubleshooting AWS identity and access or the user guide of the AWS service you are using.

Service administrator – If you're in charge of AWS resources at your company, you probably have full access to AWS. It's your job to determine which AWS features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS, see the user guide of the AWS service you are using.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS. To view example AWS identity-based policies that you can use in IAM, see the user guide of the AWS service you are using.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the

principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

  - **Service role –** A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role –** A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2 –** You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS services work with IAM

To get a high-level view of how AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

To learn how to use a specific AWS service with IAM, see the security section of the relevant service's User Guide.

Troubleshooting AWS identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS and IAM.

Topics

I am not authorized to perform an action in AWS

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `awes:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `awes:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to AWS.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my AWS resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS supports these features, see How AWS services work with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.

- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Compliance Validation for this AWS Product or Service

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- Architecting for HIPAA Security and Compliance on Amazon Web Services – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

> **Note**
>
> Not all AWS services are HIPAA eligible. For more information, see the HIPAA Eligible Services Reference.

- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.

- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

Resilience for this AWS Product or Service

The AWS global infrastructure is built around AWS Regions and Availability Zones.

AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking.

With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

Infrastructure Security for this AWS Product or Service

This AWS product or service uses managed services, and therefore is protected by the AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well‐Architected Framework*.

You use AWS published API calls to access this AWS Product or Service through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

Migrate from version 1.x to 2.x of the AWS SDK for Java

The AWS SDK for Java 2.x is a major rewrite of the 1.x code base built on top of Java 8+. It includes many updates, such as improved consistency, ease of use, and strongly enforced immutability. This section describes the major features that are new in version 2.x, and provides guidance on how to migrate your code to version 2.x from 1.x.

Topics

What’s new in version 2

- You can configure your own HTTP clients. See HTTP transport configuration.
- Async clients feature non-blocking I/O support and return `CompletableFuture` objects. See Asynchronous programming.
- Operations that return multiple pages have autopaginated responses. This way, you can focus your code on what to do with the response, without the need to check for and get subsequent pages. See Pagination.
- SDK start time performance for AWS Lambda functions is improved. See SDK start time performance improvements.
- Version 2.x supports a new shorthand method for creating requests.

  Example

  ```
  dynamoDbClient.putItem(request -> request.tableName(TABLE))
  ```

For more details about the new features and to see specific code examples, refer to the other sections of this guide.

- Quick Start

- Setting up
- Code examples for the AWS SDK for Java 2.x
- Use the SDK
- Security for the AWS SDK for Java

Migration step-by-step instructions with example

This section provides a step-by-step guide to migrate your application that currently uses the SDK for Java v1.x to the SDK for Java 2.x. The first part presents an overview of the steps followed by a detailed example of a migration.

The steps that are covered here describe a migration of a normal use case, where the application calls AWS services using model-driven service clients. If you need to migrate code that uses higher level APIs such as S3 Transfer Manager or CloudFront presigning, refer to the section under the section called “What's different between 1.x and 2.x” table of contents.

The approach described here is a suggestion. You may use other techniques and leverage your IDE's code editing features to reach the same result.

Overview of steps

1. Begin by adding the SDK for Java 2.x BOM

By adding the Maven BOM (Bill of Materials) element for the SDK for Java 2.x to your POM file, you ensure that all of the v2 dependency you need are from the same version. Your POM can contain both v1 and v2 dependencies. This allows you to incrementally migrate your code rather than change it all at once.

SDK for Java 2.x BOM

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>
      <scope>import</scope>
```

```
    </dependency>
  </dependencies>
</dependencyManagement>
```

You can find the latest version on the Maven Central Repository.

2. Search files for v1 class import statements

By scanning the files in your application for SERVICE_IDs used in v1 imports, you'll find the unique SERVICE_IDs used. A SERVICE_ID is a short, unique name for an AWS service. For example `cognitoidentity` is the SERVICE_ID for Amazon Cognito Identity.

3. Determine the v2 Maven dependencies from the v1 import statements

After you find all unique v1 SERVICE_IDs, you can determine the corresponding Maven artifact for the v2 dependency by referring to the section called “Package name to artifactId mappings”.

4. Add v2 dependency elements to the POM file

Update the Maven POM file with dependency elements determined in step 3.

5. In the Java files, incrementally change over the v1 classes to v2 classes

As you replace v1 classes with v2 classes, make the necessary changes to support the v2 API such as using builders instead of constructors and using fluent getters and setters.

6. Remove v1 Maven dependencies from the POM and v1 imports from files

After you migrate your code to use v2 classes, remove any leftover v1 imports from files and all dependencies from your build file.

7. Refactor the code to use v2 API enhancements

After the code successfully compiles and passes tests, you can take advantage of v2 enhancements such as using a different HTTP client or paginators to simplify code. This is an optional step.

Example migration

In this example, we migrate an application that uses the SDK for Java v1 and accesses several AWS services. We work through the following v1 method in detail in step 5. This is one method in a class that contains eight methods and there are 32 classes in the application.

v1 method to migrate

Only the v1 SDK imports are listed below from the Java file.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
 instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
                .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
 multiple requests.
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken());   // Prepare request for next
 page.
            for (final Reservation r : result.getReservations()) {
                for (final Instance instance : r.getInstances()) {
                    LOGGER.info("Examining instanceId: "+ instance.getInstanceId());
                    // if instance is in a running state, add it to runningInstances
 list.
                    if (RUNNING_STATES.contains(instance.getState().getName())) {
                        runningInstances.add(instance);
                    }
                }
            }
        } while (result.getNextToken() != null);
    } catch (final AmazonEC2Exception exception) {
        // if instance isn't found, assume its terminated and continue.
```

```
        if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
            LOGGER.info("Instance probably terminated; moving on.");
        } else {
            throw exception;
        }
    }
    return runningInstances;
}
```

1. Add v2 Maven BOM

Add the Maven BOM for the SDK for Java 2.x to the POM along side any other dependencies in the `dependencyManagement` section. If your POM file has the BOM for v1 of the SDK, leave it for now. It will be removed at a later step.

POM Dependency management at outset

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.example</groupId>             <!--Existing dependency in POM. -->
      <artifactId>bom</artifactId>
      <version>1.3.4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>  <!--Existing v1 BOM dependency. -->
      <version>1.11.1000</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>software.amazon.awssdk</groupId>  <!--Add v2 BOM dependency. -->
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
```

```
</dependencyManagement>
```

2. Search files for v1 class import statements

Search the application's code for unique occurrences of `import com.amazonaws.services`. This helps us determine the v1 dependencies used by the project. If your application has a Maven POM file with v1 dependencies listed, you can use this information instead.

For this example we use the [ripgrep (rg)] command to search the code base.

From the root of your code base, execute the following `ripgrep` command. After `ripgrep` finds the import statements, they are piped to the `cut`, `sort`, and `uniq` commands to isolate the SERVICE_IDs.

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

For this application, the following SERVICE_IDs are logged to the console.

```
autoscaling
cloudformation
ec2
identitymanagement
```

This indicates that there was at least one occurrence of each of the following package names used in `import` statements. Four our purposes, the individual class names don't matter. We just need to find the SERVICE_IDs that are used.

```
com.amazonaws.services.autoscaling.*
com.amazonaws.services.cloudformation.*
com.amazonaws.services.ec2.*
com.amazonaws.services.identitymanagement.*
```

3. Determine the v2 Maven dependencies from the v1 import statements

The SERVICE_IDs for v1 that we isolated from Step 2—for example `autoscaling` and `cloudformation`—can be mapped to the same v2 SERVICE_ID for the most part. Since the v2 Maven artifactId matches the SERVICE_ID in most cases, you have the information you need to add dependency blocks to your POM file.

The following table shows how we can determine the v2 dependencies.

v1 SERVICE_ID maps to ... package name	v2 SERVICE_ID maps to ... package name	v2 Maven dependency
ec2 com.amazonaws.services.**ec2**.*	**ec2** software.amazon.awssdk.services.**ec2**.*	<dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>**ec2**</artifactId> </dependency>
autoscaling com.amazonaws.services.**autoscaling**.*	**autoscaling** software.amazon.awssdk.services.**autoscaling**.*	<dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>**autoscaling**</artifactId> </dependency>
cloudformation com.amazonaws.services.**cloudformation**.*	**cloudformation** software.amazon.awssdk.**cloudformation**.*	<dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>**cloudformation**</artifactId> </dependency>
identitymanagement* com.amazonaws.services.**identitymanagement**.*	**iam*** software.amazon.awssdk.**iam**.*	<dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>**iam**</artifactId> </dependency>

* The `identitymanagement` to `iam` mapping is an exception where the SERVICE_ID differs between versions. Refer to the the section called “Package name to artifactId mappings” for exceptions if Maven or Gradle cannot resolve the v2 dependency.

4. Add v2 dependency elements to the POM file

In step 3, we determined the four dependency blocks that need to be added to the POM file. We don't need to add a version because we have specified the BOM in step 1. After the imports are added, our POM file has the following dependency elements.

```
  ...
<dependencies>
  ...
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>autoscaling</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>iam</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>cloudformation</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>ec2</artifactId>
  </dependency>
  ...
</dependencies>
  ...
```

5. In the Java files, incrementally change over the v1 classes to v2 classes

In the method that we are migrating, we see

- An EC2 service client from `com.amazonaws.services.ec2.AmazonEC2Client`.
- Several EC2 model classes used. For example `DescribeInstancesRequest` and `DescribeInstancesResult`.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
 instanceIds)
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
                .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
 multiple re
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken());   // Prepare request for next
 page.
            for (final Reservation r : result.getReservations()) {
                for (final Instance instance : r.getInstances()) {
                    LOGGER.info("Examining instanceId: "+ instance.getInstanceId());
                    // if instance is in a running state, add it to runningInstances
 list.
                    if (RUNNING_STATES.contains(instance.getState().getName())) {
                        runningInstances.add(instance);
                    }
                }
            }
        } while (result.getNextToken() != null);
    } catch (final AmazonEC2Exception exception) {
        // if instance isn't found, assume its terminated and continue.
        if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
            LOGGER.info("Instance probably terminated; moving on.");
        } else {
```

```
            throw exception;
        }
    }
    return runningInstances;
}
...
```

Our goal is to replace all v1 imports with v2 imports. We proceed one class at a time.

a. Replace import statement or class name

We see that the first parameter to the `describeRunningInstances` method is a v1 `AmazonEC2Client` instance. Do one of the following:

- Replace the import for `com.amazonaws.services.ec2.AmazonEC2Client` with `software.amazon.awssdk.services.ec2.Ec2Client` and change `AmazonEC2Client` to `Ec2Client`.
- Change the parameter type to `Ec2Client` and let the IDE prompt us for the correct import. Our IDE will prompt us to import the v2 class because the client names differ—`AmazonEC2Client` and `Ec2Client`. This approach does not work if the class name is the same in both versions.

b. Replace v1 model classes with v2 equivalents

After the change to the v2 `Ec2Client`, if we use an IDE, we see compilation errors in the following statement.

```
                    result = ec2.describeInstances(request);
```

The compilation error results from using an instance of v1's `DescribeInstancesRequest` as a parameter to the v2 `Ec2Client` `describeInstances` method. To fix, make the following replacement or import statements.

replace	with
`import com.amazonaws.services.ec2.model.DescribeInstancesRequest`	`import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest`

c. Change v1 constructors to v2 builders.

We still see compilation errors because there are no constructors on v2 classes. To fix, make the following change.

change	to
`final DescribeInstancesRequest request = new DescribeInstancesRequest() .withInstanceIds(instanceIdsCopy);`	`final DescribeInstancesRequest request = DescribeInstancesRequest.builder() .instanceIds(instanceIdsCopy) .build();`

d. Replace v1 `*Result` response objects with v2 `*Response` equivalents

A consistent difference between v1 and v2 is that all response objects in v2 end with `*Response` instead of `*Result`. Replace the v1 `DescribeInstancesResult` import to the v2 import, `DescribeInstancesResponse`.

d. Make API changes

The following statement needs a few changes.

```
request.setNextToken(result.getNextToken());
```

In v2, setter methods do not use the `set` or with `prefix`. Getter methods prefixed with `get` are also gone in the SDK for Java 2.x

Model classes, such as the `request` instance, are immutable in v2, so we need to create a new `DescribeInstancesRequest` with a builder.

In v2, the statement becomes the following.

```
request = DescribeInstancesRequest.builder()
        .nextToken(result.nextToken())
        .build();
```

d. Repeat until method compiles with v2 classes

Continue with the rest of the code. Replace v1 imports with v2 imports and fix the compilation errors. Refer to the v2 API Reference and What's different reference as needed.

After we migrate this single method, we have the following v2 code.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;

import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
...
private static List<Instance> getRunningInstances(Ec2Client ec2, List<String>
 instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
        try {
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                    .instanceIds(instanceIds)
                    .build();
            DescribeInstancesResponse result;
            do {
                // DescribeInstancesResponse is a paginated response, so use tokens
 with multiple re
                result = ec2.describeInstances(request);
                request = DescribeInstancesRequest.builder()   // Prepare request for
 next page.
                        .nextToken(result.nextToken())
                        .build();
                for (final Reservation r : result.reservations()) {
                    for (final Instance instance : r.instances()) {
                        // if instance is in a running state, add it to
 runningInstances list.
```

```
                        if (RUNNING_STATES.contains(instance.state().nameAsString())) {
                            runningInstances.add(instance);
                        }
                    }
                }
            } while (result.nextToken() != null);
        } catch (final Ec2Exception exception) {
            // if instance isn't found, assume its terminated and continue.
            if (exception.awsErrorDetails().errorCode().equals(NOT_FOUND_ERROR_CODE)) {
                    LOGGER.info("Instance probably terminated; moving on.");
            } else {
                throw exception;
            }
        }
        return runningInstances;
    }
...
```

Because we are migrating a single method in a Java file with eight methods, we have a mix of v1 and v2 imports as we work through the file. We added the last six import statements as we performed the steps.

After we migrate all the code, there will be no more v1 import statements.

6. Remove v1 Maven dependencies from the POM and v1 imports from files

After we migrate all v1 code in the file, we have the following v2 SDK import statements.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.regions.ServiceMetadata;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.InstanceStateName;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
```

After we migrate *all* files in our application, we no longer need the v1 dependencies in our POM file. Remove the v1 BOM from the `dependencyManagement` section, if using, and all v1 dependency blocks.

7. Refactor the code to use v2 API enhancements

For the snippet we have been migrating, we can optionally use a v2 paginator and let the SDK manage the token-based requests for more data.

We can replace the entire do clause with the following.

```
                DescribeInstancesIterable responses =
ec2.describeInstancesPaginator(request);

                responses.reservations().stream()
                        .forEach(reservation -> reservation.instances()
                                .forEach(instance -> {
                                    if
(RUNNING_STATES.contains(instance.state().nameAsString())) {
                                        runningInstances.put(instance.instanceId(),
instance);
                                    }
                                }));
```

Package name to Maven artifactId mappings

When you migrate your Maven or Gradle project from v1 of the SDK for Java to v2, you need to figure out which dependencies to add to your build file. The approach described in the [the section called “Step-by-step instructions”](#) (step 3) uses the package names in import statements as a starting point to determine the dependencies (as artifactIds) to add to your build file.

You can use the information in this topic to map the v1 package names to v2 artifactIds.

Common naming convention used in package names and Maven artifactIds

The following table shows the common naming convention that the SDKs use for a given SERVICE_ID. A SERVICE_ID is a unique identifier for an AWS service. For example, the SERVICE_ID for the Amazon S3 service is `s3` and `cognitoidentity` is the SERVICE_ID for Amazon Cognito Identity.

v1 package name (import statement)	v1 artifactId	v2 artifactId	v2 package name (import statement)
com.amazonaws.services.SERVICE_ID	aws-java-sdk-SERVICE_ID	SERVICE_ID	software.amazon.awssdk.services.SERVICE_ID
Example for Amazon Cognito Identity (SERVICE_ID: `cognitoidentity`)			
com.amazonaws.services.***cognitoidentity***	aws-java-sdk-***cognitoidentity***	***cognitoidentity***	software.amazon.awssdk.services.***cognitoidentity***

SERVICE_ID differences

Within v1

In some cases the SERVICE_ID differs between the package name and in the artifactId for the same service. For example, the CloudWatch Metrics row of the following table shows that `metrics` is the SERVICE_ID in the package name but `cloudwatchmetrics` is the artifactId's SERVICE_ID.

Within v2

There are no differences in the SERVICE_ID used in package names and artifactIds.

Between v1 and v2

For the majority of services, the SERVICE_ID in v2 is the same as v1's SERVICE_ID in both package names and artifactIds. An example of this is the `cognitoedentity` SERVICE_ID as shown in the previous table. However, some SERVICE_IDs differ between the SDKs as shown in the following table.

A **boldface SERVICE_ID** in either of the v1 columns indicates that it's different from the SERVICE_ID used in v2.

Service name	v1 package name	v1 artifactId	v2 artifactId	v2 package name
	All package names begin with `com.amazonaws.services` as shown in the first row.	All artifactIds are enclosed in tags as shown in the first row.	All artifactIds are enclosed in tags as shown in the first row.	All package names begin with `software.amazon.awssdk` as shown in the first row.
API Gateway	com.amazonaws.services.apigateway	<artifactId>aws-java-sdk-**api-gateway**</artifactId>	<artifactId>apigateway</artifactId>	software.amazon.awssdk.services.apigateway
App Registry	**appregistry**	**appregistry**	servicecatalogappregistry	servicecatalogappregistry
Application Discovery	applicationdiscovery	**discovery**	applicationdiscovery	applicationdiscovery
Augmented AI Runtime	**augmentedairuntime**	**augmentedairuntime**	sagemakera2iruntime	sagemakera2iruntime
Certificate Manager	**certificatemanager**	acm	acm	acm
CloudControl API	**cloudcontrolapi**	cloudcontrolapi	cloudcontrol	cloudcontrol
CloudSearch	**cloudsearchv2**	cloudsearch	cloudsearch	cloudsearch
CloudSearch Domain	cloudsearchdomain	**cloudsearch**	cloudsearchdomain	cloudsearchdomain
CloudWatch Events	cloudwatchevents	**events**	cloudwatchevents	cloudwatchevents

Service name	v1 package name	v1 artifactId	v2 artifactId	v2 package name
CloudWatch Evidently	**cloudwatchevidently**	**cloudwatchevidently**	evidently	evidently
CloudWatch Logs	**logs**	**logs**	cloudwatchlogs	cloudwatchlogs
CloudWatch Metrics	**metrics**	**cloudwatchmetrics**	cloudwatch	cloudwatch
CloudWatch Rum	**cloudwatchrum**	**cloudwatchrum**	rum	rum
Cognito Identity Provider	**cognitoidp**	**cognitoidp**	cognitoidentityprovider	cognitoidentityprovider
Connect Campaign	**connectcampaign**	**connectcampaign**	connectcampaigns	connectcampaigns
Connect Wisdom	**connectwisdom**	**connectwisdom**	wisdom	wisdom
Database Migration Service	**databasemigrationservice**	**dms**	databasemigration	databasemigration
DataZone	datazone	**datazoneexternal**	datazone	datazone
DynamoDB	**dynamodbv2**	dynamodb	dynamodb	dynamodb
Elastic File System	**elasticfilesystem**	efs	efs	efs
Elastic Map Reduce	**elasticmapreduce**	emr	emr	emr
Glue DataBrew	**gluedatabrew**	**gluedatabrew**	databrew	databrew

Service name	v1 package name	v1 artifactId	v2 artifactId	v2 package name
IAM Roles Anywhere	**iamrolesa nywhere**	**iamrolesa nywhere**	rolesanywhere	rolesanywhere
Identity Management	**identitym anagement**	iam	iam	iam
IoT Data	**iotdata**	**iot**	iotdataplane	iotdataplane
Kinesis Analytics	kinesisanalytics	**kinesis**	kinesisanalytics	kinesisanalytics
Kinesis Firehose	**kinesisfirehose**	**kinesis**	firehose	firehose
Kinesis Video Signaling Channels	**kinesisvi deosignal ingchannels**	**kinesisvi deosignal ingchannels**	kinesisvi deosignaling	kinesisvi deosignaling
Lex	lexruntime	**lex**	lexruntime	lexruntime
Lookout For Vision	**lookoutfo rvision**	**lookoutfo rvision**	lookoutvision	lookoutvision
Mainframe Modernization	**mainframe modernization**	**mainframe modernization**	m2	m2
Marketplace Metering	marketpla cemetering	**marketpla cemeterin gservice**	marketpla cemetering	marketpla cemetering
Managed Grafana	**managedgr afana**	**managedgr afana**	grafana	grafana
Mechanical Turk	mturk	**mechanica lturkrequester**	mturk	mturk

Service name	v1 package name	v1 artifactId	v2 artifactId	v2 package name
Migration Hub Strategy Recommendations	**migrationhubstrategyrecommendations**	**migrationhubstrategyrecommendations**	migrationhubstrategy	migrationhubstrategy
Nimble Studio	**nimblestudio**	**nimblestudio**	nimble	nimble
Private 5G	**private5g**	**private5g**	privatenetworks	privatenetworks
Prometheus	**prometheus**	**prometheus**	amp	amp
Recycle Bin	**recyclebin**	**recyclebin**	rbin	rbin
Redshift Data API	**redshiftdataapi**	**redshiftdataapi**	redshiftdata	redshiftdata
Route 53	route53domains	**route53**	route53domains	route53domains
Sage Maker Edge Manager	**sagemakeredgemanager**	**sagemakeredgemanager**	sagemakeredge	sagemakeredge
Security Token	**securitytoken**	sts	sts	sts
Server Migration	**servermigration**	**servermigration**	sms	sms
Simple Email	**simpleemail**	ses	ses	ses
Simple Email V2	**simpleemailv2**	sesv2	sesv2	sesv2
Simple Systems Management	**simplesystemsmanagement**	ssm	ssm	ssm
Simple Workflow	**simpleworkflow**	**simpleworkflow**	swf	swf
Step Functions	**stepfunctions**	**stepfunctions**	sfn	sfn

What's different between the AWS SDK for Java 1.x and 2.x

This section describes the main changes to be aware of when converting an application from using the AWS SDK for Java version 1.x to version 2.x.

Package name change

A noticeable change from the SDK for Java 1.x to the SDK for Java 2.x is the package name change. Package names begin with `software.amazon.awssdk` in SDK 2.x, whereas the SDK 1.x uses `com.amazonaws`.

These same names differentiate Maven artifacts from SDK 1.x to SDK 2.x. Maven artifacts for the SDK 2.x use the `software.amazon.awssdk` groupId, whereas the SDK 1.x uses the `com.amazonaws` groupId.

There are a few times when your code requires a `com.amazonaws` dependency for a project that otherwise uses only SDK 2.x artifacts. One example of this is when you work with server-side AWS Lambda. This was shown in the Set up an Apache Maven project section earlier in this guide.

> **Note**
>
> Several package names in the SDK 1.x contain `v2`. The use of `v2` in this case usually means that code in the package is targeted to work with version 2 of the service.
> Since the full package name begins with `com.amazonaws`, these are SDK 1.x components. Examples of these package names in the SDK 1.x are:
>
> - `com.amazonaws.services.dynamodbv2`
> - `com.amazonaws.retry.v2`
> - `com.amazonaws.services.apigatewayv2`
> - `com.amazonaws.services.simpleemailv2`

Adding version 2.x to your project

Maven is the recommended way to manage dependencies when using the AWS SDK for Java 2.x. To add version 2.x components to your project, update your `pom.xml` file with a dependency on the SDK.

Example

```
<dependencyManagement>
    <dependencies>
        <dependency>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>bom</artifactId>
          <version>2.16.1</version>
          <type>pom</type>
          <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb</artifactId>
    </dependency>
</dependencies>
```

You can also use version 1.x and 2.x side-by-side as you migrate your project to version 2.x.

Immutable POJOs

Clients and operation request and response objects are now immutable and cannot be changed after creation. To reuse a request or response variable, you must build a new object to assign to it.

Example of updating a request object in 1.x

```
DescribeAlarmsRequest request = new DescribeAlarmsRequest();
DescribeAlarmsResult response = cw.describeAlarms(request);

request.setNextToken(response.getNextToken());
```

Example of updating a request object in 2.x

```
DescribeAlarmsRequest request = DescribeAlarmsRequest.builder().build();
DescribeAlarmsResponse response = cw.describeAlarms(request);

request = DescribeAlarmsRequest.builder()
        .nextToken(response.nextToken())
```

```
        .build();
```

Setter and getter methods

In the AWS SDK for Java 2.x, setter method names don't include the `set` or `with` prefix. For example, `*.withEndpoint()` is now `*.endpoint()`.

Getter method names do not use the `get` prefix.

Example of using setter methods in 1.x

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
          .withRegion("us-east-1")
          .build();
```

Example of using setter methods in 2.x

```
DynamoDbClient client = DynamoDbClient.builder()
          .region(Region.US_EAST_1)
          .build();
```

Example of using getter methods in 1.x

```
String token = request.getNextToken();
```

Example of using getter methods in 2.x

```
String token = request.nextToken();
```

Model class names

Model class names that represent service responses end with `Response` in v2 instead of `Result` that v1 uses.

Example of class names that represent a response in v1

```
CreateApiKeyResult
AllocateAddressResult
```

Example of class names that represent a response in v2

```
CreateApiKeyResponse
AllocateAddressResponse
```

Migration status of libraries and utilities

SDK for Java libraries and utilities

The following table lists the migration status of libraries and utilities for the SDK for Java.

Version 1.12.x name	Version 2.x name	Since version in 2.x
DynamoDBMapper	DynamoDbEnhancedClient	2.12.0
Waiters	Waiters	2.15.0
CloudFrontUrlSigner, CloudFrontCookieSigner	CloudFrontUtilities	2.18.33
TransferManager	S3TransferManager	2.19.0
EC2 Metadata client	EC2 Metadata client	2.19.29
S3 URI parser	S3 URI parser	2.20.41
IAM Policy Builder	IAM Policy Builder	2.20.126
Amazon SQS Client-side Buffering	Automatic Request Batching	not yet released
Progress Listeners	Progress Listeners	not yet released

Related libraries

The following table lists libraries that are released separately but work with the SDK for Java 2.x.

Name used with version 2.x of the SDK for Java	Since version
Amazon S3 Encryption Client	3.0.0[1]
AWS Database Encryption Client for DynamoDB	3.0.0[2]

[1]The encryption client for Amazon S3 is available by using the following Maven dependency.

```
<dependency>
    <groupId>software.amazon.encryption.s3</groupId>
    <artifactId>amazon-s3-encryption-client-java</artifactId>
    <version>3.x</version>
</dependency>
```

[2]The AWS Database Encryption Client for DynamoDB is available by using the following Maven dependency.

```
<dependency>
    <groupId>software.amazon.cryptography</groupId>
    <artifactId>aws-database-encryption-sdk-dynamodb</artifactId>
    <version>3.x</version>
</dependency>
```

Migration details for libraries and utilities

- S3 Transfer Manager
- EC2 metadata utility
- CloudFront presigning
- S3 URI parsing

Client changes

Client builders

You must create all clients using the client builder method. Constructors are no longer available.

Example of creating a client in version 1.x

```
AmazonDynamoDB ddbClient = AmazonDynamoDBClientBuilder.defaultClient();
AmazonDynamoDBClient ddbClient = new AmazonDynamoDBClient();
```

Example of creating a client in version 2.x

```
DynamoDbClient ddbClient = DynamoDbClient.create();
DynamoDbClient ddbClient = DynamoDbClient.builder().build();
```

Client class names

All client class names are now fully camel cased and no longer prefixed by `Amazon`. These changes are aligned with names used in the AWS CLI.

Example of class names in 1.x

```
AmazonDynamoDB
AWSACMPCAAsyncClient
```

Example of class names in 2.x

```
DynamoDbClient
AcmAsyncClient
```

Client class name changes

1.x Client	2.x Client
`com.amazonaws.services.acmpca.AWSACMPCAAsyncClient`	`software.amazon.awssdk.services.acm.AcmAsyncClient`
`com.amazonaws.services.acmpca.AWSACMPCAClient`	`software.amazon.awssdk.services.acm.AcmClient`
`com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessAsyncClient`	`software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessAsyncClient`

1.x Client	2.x Client
com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessClient	software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessClient
com.amazonaws.services.apigateway.AmazonApiGatewayAsyncClient	software.amazon.awssdk.services.apigateway.ApiGatewayAsyncClient
com.amazonaws.services.apigateway.AmazonApiGatewayClient	software.amazon.awssdk.services.apigateway.ApiGatewayClient
com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingAsyncClient	software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingAsyncClient
com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClient	software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient
com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryAsyncClient	software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryAsyncClient
com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryClient	software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryClient
com.amazonaws.services.appstream.AmazonAppStreamAsyncClient	software.amazon.awssdk.services.appstream.AppStreamAsyncClient
com.amazonaws.services.appstream.AmazonAppStreamClient	software.amazon.awssdk.services.appstream.AppStreamClient

1.x Client	2.x Client
com.amazonaws.services.appsync.AWSAppSyncAsyncClient	software.amazon.awssdk.services.appsync.AppSyncAsyncClient
com.amazonaws.services.appsync.AWSAppSyncClient	software.amazon.awssdk.services.appsync.AppSyncClient
com.amazonaws.services.athena.AmazonAthenaAsyncClient	software.amazon.awssdk.services.athena.AthenaAsyncClient
com.amazonaws.services.athena.AmazonAthenaClient	software.amazon.awssdk.services.athena.AthenaClient
com.amazonaws.services.autoscaling.AmazonAutoScalingAsyncClient	software.amazon.awssdk.services.autoscaling.AutoScalingAsyncClient
com.amazonaws.services.autoscaling.AmazonAutoScalingClient	software.amazon.awssdk.services.autoscaling.AutoScalingClient
com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansAsyncClient	software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansAsyncClient
com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansClient	software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansClient
com.amazonaws.services.batch.AWSBatchAsyncClient	software.amazon.awssdk.services.batch.BatchAsyncClient
com.amazonaws.services.batch.AWSBatchClient	software.amazon.awssdk.services.batch.BatchClient
com.amazonaws.services.budgets.AWSBudgetsAsyncClient	software.amazon.awssdk.services.budgets.BudgetsAsyncClient

1.x Client	2.x Client
com.amazonaws.services.budgets.AWSBudgetsClient	software.amazon.awssdk.services.budgets.BudgetsClient
com.amazonaws.services.certificatemanager.AWSCertificateManagerAsyncClient	software.amazon.awssdk.services.acm.AcmAsyncClient
com.amazonaws.services.certificatemanager.AWSCertificateManagerClient	software.amazon.awssdk.services.acm.AcmClient
com.amazonaws.services.cloud9.AWSCloud9AsyncClient	software.amazon.awssdk.services.cloud9.Cloud9AsyncClient
com.amazonaws.services.cloud9.AWSCloud9Client	software.amazon.awssdk.services.cloud9.Cloud9Client
com.amazonaws.services.clouddirectory.AmazonCloudDirectoryAsyncClient	software.amazon.awssdk.services.clouddirectory.CloudDirectoryAsyncClient
com.amazonaws.services.clouddirectory.AmazonCloudDirectoryClient	software.amazon.awssdk.services.clouddirectory.CloudDirectoryClient
com.amazonaws.services.cloudformation.AmazonCloudFormationAsyncClient	software.amazon.awssdk.services.cloudformation.CloudFormationAsyncClient
com.amazonaws.services.cloudformation.AmazonCloudFormationClient	software.amazon.awssdk.services.cloudformation.CloudFormationClient
com.amazonaws.services.cloudfront.AmazonCloudFrontAsyncClient	software.amazon.awssdk.services.cloudfront.CloudFrontAsyncClient

1.x Client	2.x Client
com.amazonaws.services.cloudfront.AmazonCloudFrontClient	software.amazon.awssdk.services.cloudfront.CloudFrontClient
com.amazonaws.services.cloudhsm.AWSCloudHSMAsyncClient	software.amazon.awssdk.services.cloudhsm.CloudHsmAsyncClient
com.amazonaws.services.cloudhsm.AWSCloudHSMClient	software.amazon.awssdk.services.cloudhsm.CloudHsmClient
com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2AsyncClient	software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2AsyncClient
com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2Client	software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2Client
com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainAsyncClient	software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainAsyncClient
com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainClient	software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainClient
com.amazonaws.services.cloudsearchv2.AmazonCloudSearchAsyncClient	software.amazon.awssdk.services.cloudsearch.CloudSearchAsyncClient
com.amazonaws.services.cloudsearchv2.AmazonCloudSearchClient	software.amazon.awssdk.services.cloudsearch.CloudSearchClient

1.x Client	2.x Client
com.amazonaws.services.cloudtrail.AWSCloudTrailAsyncClient	software.amazon.awssdk.services.cloudtrail.CloudTrailAsyncClient
com.amazonaws.services.cloudtrail.AWSCloudTrailClient	software.amazon.awssdk.services.cloudtrail.CloudTrailClient
com.amazonaws.services.cloudwatch.AmazonCloudWatchAsyncClient	software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient
com.amazonaws.services.cloudwatch.AmazonCloudWatchClient	software.amazon.awssdk.services.cloudwatch.CloudWatchClient
com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsAsyncClient	software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsAsyncClient
com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClient	software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient
com.amazonaws.services.codebuild.AWSCodeBuildAsyncClient	software.amazon.awssdk.services.codebuild.CodeBuildAsyncClient
com.amazonaws.services.codebuild.AWSCodeBuildClient	software.amazon.awssdk.services.codebuild.CodeBuildClient
com.amazonaws.services.codecommit.AWSCodeCommitAsyncClient	software.amazon.awssdk.services.codecommit.CodeCommitAsyncClient

1.x Client	2.x Client
com.amazonaws.services.codecommit.AWSCodeCommitClient	software.amazon.awssdk.services.codecommit.CodeCommitClient
com.amazonaws.services.codedeploy.AmazonCodeDeployAsyncClient	software.amazon.awssdk.services.codedeploy.CodeDeployAsyncClient
com.amazonaws.services.codedeploy.AmazonCodeDeployClient	software.amazon.awssdk.services.codedeploy.CodeDeployClient
com.amazonaws.services.codepipeline.AWSCodePipelineAsyncClient	software.amazon.awssdk.services.codepipeline.CodePipelineAsyncClient
com.amazonaws.services.codepipeline.AWSCodePipelineClient	software.amazon.awssdk.services.codepipeline.CodePipelineClient
com.amazonaws.services.codestar.AWSCodeStarAsyncClient	software.amazon.awssdk.services.codestar.CodeStarAsyncClient
com.amazonaws.services.codestar.AWSCodeStarClient	software.amazon.awssdk.services.codestar.CodeStarClient
com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityAsyncClient	software.amazon.awssdk.services.cognitoidentity.CognitoIdentityAsyncClient
com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityClient	software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient

1.x Client	2.x Client
com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderAsyncClient	software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderAsyncClient
com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderClient	software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient
com.amazonaws.services.cognitosync.AmazonCognitoSyncAsyncClient	software.amazon.awssdk.services.cognitosync.CognitoSyncAsyncClient
com.amazonaws.services.cognitosync.AmazonCognitoSyncClient	software.amazon.awssdk.services.cognitosync.CognitoSyncClient
com.amazonaws.services.comprehend.AmazonComprehendAsyncClient	software.amazon.awssdk.services.comprehend.ComprehendAsyncClient
com.amazonaws.services.comprehend.AmazonComprehendClient	software.amazon.awssdk.services.comprehend.ComprehendClient
com.amazonaws.services.config.AmazonConfigAsyncClient	software.amazon.awssdk.services.config.ConfigAsyncClient
com.amazonaws.services.config.AmazonConfigClient	software.amazon.awssdk.services.config.ConfigClient
com.amazonaws.services.connect.AmazonConnectAsyncClient	software.amazon.awssdk.services.connect.ConnectAsyncClient
com.amazonaws.services.connect.AmazonConnectClient	software.amazon.awssdk.services.connect.ConnectClient

1.x Client	2.x Client
com.amazonaws.services.costandusagereport.AWSCostAndUsageReportAsyncClient	software.amazon.awssdk.services.costandusagereport.CostAndUsageReportAsyncClient
com.amazonaws.services.costandusagereport.AWSCostAndUsageReportClient	software.amazon.awssdk.services.costandusagereport.CostAndUsageReportClient
com.amazonaws.services.costexplorer.AWSCostExplorerAsyncClient	software.amazon.awssdk.services.costexplorer.CostExplorerAsyncClient
com.amazonaws.services.costexplorer.AWSCostExplorerClient	software.amazon.awssdk.services.costexplorer.CostExplorerClient
com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceAsyncClient	software.amazon.awssdk.services.databasemigration.DatabaseMigrationAsyncClient
com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceClient	software.amazon.awssdk.services.databasemigration.DatabaseMigrationClient
com.amazonaws.services.datapipeline.DataPipelineAsyncClient	software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient
com.amazonaws.services.datapipeline.DataPipelineClient	software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient
com.amazonaws.services.dax.AmazonDaxAsyncClient	software.amazon.awssdk.services.dax.DaxAsyncClient

1.x Client	2.x Client
com.amazonaws.services.dax.AmazonDaxClient	software.amazon.awssdk.services.dax.DaxClient
com.amazonaws.services.devicefarm.AWSDeviceFarmAsyncClient	software.amazon.awssdk.services.devicefarm.DeviceFarmAsyncClient
com.amazonaws.services.devicefarm.AWSDeviceFarmClient	software.amazon.awssdk.services.devicefarm.DeviceFarmClient
com.amazonaws.services.directconnect.AmazonDirectConnectAsyncClient	software.amazon.awssdk.services.directconnect.DirectConnectAsyncClient
com.amazonaws.services.directconnect.AmazonDirectConnectClient	software.amazon.awssdk.services.directconnect.DirectConnectClient
com.amazonaws.services.directory.AWSDirectoryServiceAsyncClient	software.amazon.awssdk.services.directory.DirectoryAsyncClient
com.amazonaws.services.directory.AWSDirectoryServiceClient	software.amazon.awssdk.services.directory.DirectoryClient
com.amazonaws.services.dlm.AmazonDLMAsyncClient	software.amazon.awssdk.services.dlm.DlmAsyncClient
com.amazonaws.services.dlm.AmazonDLMClient	software.amazon.awssdk.services.dlm.DlmClient
com.amazonaws.services.dynamodbv2.AmazonDynamoDBAsyncClient	software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient

1.x Client	2.x Client
com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient	software.amazon.awssdk.services.dynamodb.DynamoDbClient
com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsAsyncClient	software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsAsyncClient
com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClient	software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsClient
com.amazonaws.services.ec2.AmazonEC2AsyncClient	software.amazon.awssdk.services.ec2.Ec2AsyncClient
com.amazonaws.services.ec2.AmazonEC2Client	software.amazon.awssdk.services.ec2.Ec2Client
com.amazonaws.services.ecr.AmazonECRAsyncClient	software.amazon.awssdk.services.ecr.EcrAsyncClient
com.amazonaws.services.ecr.AmazonECRClient	software.amazon.awssdk.services.ecr.EcrClient
com.amazonaws.services.ecs.AmazonECSAsyncClient	software.amazon.awssdk.services.ecs.EcsAsyncClient
com.amazonaws.services.ecs.AmazonECSClient	software.amazon.awssdk.services.ecs.EcsClient
com.amazonaws.services.eks.AmazonEKSAsyncClient	software.amazon.awssdk.services.eks.EksAsyncClient
com.amazonaws.services.eks.AmazonEKSClient	software.amazon.awssdk.services.eks.EksClient

1.x Client	2.x Client
com.amazonaws.services.elasticache.AmazonElastiCacheAsyncClient	software.amazon.awssdk.services.elasticache.ElastiCacheAsyncClient
com.amazonaws.services.elasticache.AmazonElastiCacheClient	software.amazon.awssdk.services.elasticache.ElastiCacheClient
com.amazonaws.services.elasticbeanstalk.AWSElasticBeanstalkAsyncClient	software.amazon.awssdk.services.elasticbeanstalk.ElasticBeanstalkAsyncClient
com.amazonaws.services.elasticbeanstalk.AWSElasticBeanstalkClient	software.amazon.awssdk.services.elasticbeanstalk.ElasticBeanstalkClient
com.amazonaws.services.elasticfilesystem.AmazonElasticFileSystemAsyncClient	software.amazon.awssdk.services.efs.EfsAsyncClient
com.amazonaws.services.elasticfilesystem.AmazonElasticFileSystemClient	software.amazon.awssdk.services.efs.EfsClient
com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingAsyncClient	software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingAsyncClient
com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingClient	software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingClient
com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingAsyncClient	software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2AsyncClient

1.x Client	2.x Client
com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingClient	software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2Client
com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceAsyncClient	software.amazon.awssdk.services.emr.EmrAsyncClient
com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClient	software.amazon.awssdk.services.emr.EmrClient
com.amazonaws.services.elasticsearch.AWSElasticsearchAsyncClient	software.amazon.awssdk.services.elasticsearch.ElasticsearchAsyncClient
com.amazonaws.services.elasticsearch.AWSElasticsearchClient	software.amazon.awssdk.services.elasticsearch.ElasticsearchClient
com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderAsyncClient	software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderAsyncClient
com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderClient	software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderClient
com.amazonaws.services.fms.AWSFMSAsyncClient	software.amazon.awssdk.services.fms.FmsAsyncClient
com.amazonaws.services.fms.AWSFMSClient	software.amazon.awssdk.services.fms.FmsClient

1.x Client	2.x Client
com.amazonaws.services.gamelift.AmazonGameLiftAsyncClient	software.amazon.awssdk.services.gamelift.GameLiftAsyncClient
com.amazonaws.services.gamelift.AmazonGameLiftClient	software.amazon.awssdk.services.gamelift.GameLiftClient
com.amazonaws.services.glacier.AmazonGlacierAsyncClient	software.amazon.awssdk.services.glacier.GlacierAsyncClient
com.amazonaws.services.glacier.AmazonGlacierClient	software.amazon.awssdk.services.glacier.GlacierClient
com.amazonaws.services.glue.AWSGlueAsyncClient	software.amazon.awssdk.services.glue.GlueAsyncClient
com.amazonaws.services.glue.AWSGlueClient	software.amazon.awssdk.services.glue.GlueClient
com.amazonaws.services.greengrass.AWSGreengrassAsyncClient	software.amazon.awssdk.services.greengrass.GreengrassAsyncClient
com.amazonaws.services.greengrass.AWSGreengrassClient	software.amazon.awssdk.services.greengrass.GreengrassClient
com.amazonaws.services.guardduty.AmazonGuardDutyAsyncClient	software.amazon.awssdk.services.guardduty.GuardDutyAsyncClient
com.amazonaws.services.guardduty.AmazonGuardDutyClient	software.amazon.awssdk.services.guardduty.GuardDutyClient
com.amazonaws.services.health.AWSHealthAsyncClient	software.amazon.awssdk.services.health.HealthAsyncClient

1.x Client	2.x Client
com.amazonaws.services.health.AWSHealthClient	software.amazon.awssdk.services.health.HealthClient
com.amazonaws.services.identitymanagement.AmazonIdentityManagementAsyncClient	software.amazon.awssdk.services.iam.IamAsyncClient
com.amazonaws.services.identitymanagement.AmazonIdentityManagementClient	software.amazon.awssdk.services.iam.IamClient
com.amazonaws.services.importexport.AmazonImportExportAsyncClient	software.amazon.awssdk.services.importexport.ImportExportAsyncClient
com.amazonaws.services.importexport.AmazonImportExportClient	software.amazon.awssdk.services.importexport.ImportExportClient
com.amazonaws.services.inspector.AmazonInspectorAsyncClient	software.amazon.awssdk.services.inspector.InspectorAsyncClient
com.amazonaws.services.inspector.AmazonInspectorClient	software.amazon.awssdk.services.inspector.InspectorClient
com.amazonaws.services.iot.AWSIotAsyncClient	software.amazon.awssdk.services.iot.IotAsyncClient
com.amazonaws.services.iot.AWSIotClient	software.amazon.awssdk.services.iot.IotClient
com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesAsyncClient	software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesAsyncClient

1.x Client	2.x Client
com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesClient	software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesClient
com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsAsyncClient	software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsAsyncClient
com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsClient	software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsClient
com.amazonaws.services.iotanalytics.AWSIoTAnalyticsAsyncClient	software.amazon.awssdk.services.iotanalytics.IotAnalyticsAsyncClient
com.amazonaws.services.iotanalytics.AWSIoTAnalyticsClient	software.amazon.awssdk.services.iotanalytics.IotAnalyticsClient
com.amazonaws.services.iotdata.AWSIotDataAsyncClient	software.amazon.awssdk.services.iotdata.IotDataAsyncClient
com.amazonaws.services.iotdata.AWSIotDataClient	software.amazon.awssdk.services.iotdata.IotDataClient
com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneAsyncClient	software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient
com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneClient	software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient
com.amazonaws.services.kinesis.AmazonKinesisAsyncClient	software.amazon.awssdk.services.kinesis.KinesisAsyncClient

1.x Client	2.x Client
com.amazonaws.services.kinesis.AmazonKinesisClient	software.amazon.awssdk.services.kinesis.KinesisClient
com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsAsyncClient	software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsAsyncClient
com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsClient	software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsClient
com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseAsyncClient	software.amazon.awssdk.services.firehose.FirehoseAsyncClient
com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseClient	software.amazon.awssdk.services.firehose.FirehoseClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaAsyncClient	software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaAsyncClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaClient	software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoAsyncClient	software.amazon.awssdk.services.kinesisvideo.KinesisVideoAsyncClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoClient	software.amazon.awssdk.services.kinesisvideo.KinesisVideoClient

1.x Client	2.x Client
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaAsyncClient	software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaAsyncClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaClient	software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoPutMediaClient	Not Supported
com.amazonaws.services.kms.AWSKMSAsyncClient	software.amazon.awssdk.services.kms.KmsAsyncClient
com.amazonaws.services.kms.AWSKMSClient	software.amazon.awssdk.services.kms.KmsClient
com.amazonaws.services.lambda.AWSLambdaAsyncClient	software.amazon.awssdk.services.lambda.LambdaAsyncClient
com.amazonaws.services.lambda.AWSLambdaClient	software.amazon.awssdk.services.lambda.LambdaClient
com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingAsyncClient	software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingAsyncClient
com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingClient	software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingClient
com.amazonaws.services.lexruntime.AmazonLexRuntimeAsyncClient	software.amazon.awssdk.services.lexruntime.LexRuntimeAsyncClient

1.x Client	2.x Client
com.amazonaws.services.lexruntime.AmazonLexRuntimeClient	software.amazon.awssdk.services.lexruntime.LexRuntimeClient
com.amazonaws.services.lightsail.AmazonLightsailAsyncClient	software.amazon.awssdk.services.lightsail.LightsailAsyncClient
com.amazonaws.services.lightsail.AmazonLightsailClient	software.amazon.awssdk.services.lightsail.LightsailClient
com.amazonaws.services.logs.AWSLogsAsyncClient	software.amazon.awssdk.services.logs.LogsAsyncClient
com.amazonaws.services.logs.AWSLogsClient	software.amazon.awssdk.services.logs.LogsClient
com.amazonaws.services.machinelearning.AmazonMachineLearningAsyncClient	software.amazon.awssdk.services.machinelearning.MachineLearningAsyncClient
com.amazonaws.services.machinelearning.AmazonMachineLearningClient	software.amazon.awssdk.services.machinelearning.MachineLearningClient
com.amazonaws.services.macie.AmazonMacieAsyncClient	software.amazon.awssdk.services.macie.MacieAsyncClient
com.amazonaws.services.macie.AmazonMacieClient	software.amazon.awssdk.services.macie.MacieClient
com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsAsyncClient	software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsAsyncClient

1.x Client	2.x Client
com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient	software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsClient
com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementAsyncClient	software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementAsyncClient
com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementClient	software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementClient
com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringAsyncClient	software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringAsyncClient
com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClient	software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringClient
com.amazonaws.services.mediaconvert.AWSMediaConvertAsyncClient	software.amazon.awssdk.services.mediaconvert.MediaConvertAsyncClient
com.amazonaws.services.mediaconvert.AWSMediaConvertClient	software.amazon.awssdk.services.mediaconvert.MediaConvertClient
com.amazonaws.services.medialive.AWSMediaLiveAsyncClient	software.amazon.awssdk.services.medialive.MediaLiveAsyncClient
com.amazonaws.services.medialive.AWSMediaLiveClient	software.amazon.awssdk.services.medialive.MediaLiveClient

1.x Client	2.x Client
com.amazonaws.services.mediapackage.AWSMediaPackageAsyncClient	software.amazon.awssdk.services.mediapackage.MediaPackageAsyncClient
com.amazonaws.services.mediapackage.AWSMediaPackageClient	software.amazon.awssdk.services.mediapackage.MediaPackageClient
com.amazonaws.services.mediastore.AWSMediaStoreAsyncClient	software.amazon.awssdk.services.mediastore.MediaStoreAsyncClient
com.amazonaws.services.mediastore.AWSMediaStoreClient	software.amazon.awssdk.services.mediastore.MediaStoreClient
com.amazonaws.services.mediastoredata.AWSMediaStoreDataAsyncClient	software.amazon.awssdk.services.mediastoredata.MediaStoreDataAsyncClient
com.amazonaws.services.mediastoredata.AWSMediaStoreDataClient	software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient
com.amazonaws.services.mediatailor.AWSMediaTailorAsyncClient	software.amazon.awssdk.services.mediatailor.MediaTailorAsyncClient
com.amazonaws.services.mediatailor.AWSMediaTailorClient	software.amazon.awssdk.services.mediatailor.MediaTailorClient
com.amazonaws.services.migrationhub.AWSMigrationHubAsyncClient	software.amazon.awssdk.services.migrationhub.MigrationHubAsyncClient

1.x Client	2.x Client
com.amazonaws.services.migrationhub.AWSMigrationHubClient	software.amazon.awssdk.services.migrationhub.MigrationHubClient
com.amazonaws.services.mobile.AWSMobileAsyncClient	software.amazon.awssdk.services.mobile.MobileAsyncClient
com.amazonaws.services.mobile.AWSMobileClient	software.amazon.awssdk.services.mobile.MobileClient
com.amazonaws.services.mq.AmazonMQAsyncClient	software.amazon.awssdk.services.mq.MqAsyncClient
com.amazonaws.services.mq.AmazonMQClient	software.amazon.awssdk.services.mq.MqClient
com.amazonaws.services.mturk.AmazonMTurkAsyncClient	software.amazon.awssdk.services.mturk.MTurkAsyncClient
com.amazonaws.services.mturk.AmazonMTurkClient	software.amazon.awssdk.services.mturk.MTurkClient
com.amazonaws.services.neptune.AmazonNeptuneAsyncClient	software.amazon.awssdk.services.neptune.NeptuneAsyncClient
com.amazonaws.services.neptune.AmazonNeptuneClient	software.amazon.awssdk.services.neptune.NeptuneClient
com.amazonaws.services.opsworks.AWSOpsWorksAsyncClient	software.amazon.awssdk.services.opsworks.OpsWorksAsyncClient
com.amazonaws.services.opsworks.AWSOpsWorksClient	software.amazon.awssdk.services.opsworks.OpsWorksClient

1.x Client	2.x Client
com.amazonaws.services.opsworkscm.AWSOpsWorksCMAsyncClient	software.amazon.awssdk.services.opsworkscm.OpsWorksCmAsyncClient
com.amazonaws.services.opsworkscm.AWSOpsWorksCMClient	software.amazon.awssdk.services.opsworkscm.OpsWorksCmClient
com.amazonaws.services.organizations.AWSOrganizationsAsyncClient	software.amazon.awssdk.services.organizations.OrganizationsAsyncClient
com.amazonaws.services.organizations.AWSOrganizationsClient	software.amazon.awssdk.services.organizations.OrganizationsClient
com.amazonaws.services.pi.AWSPIAsyncClient	software.amazon.awssdk.services.pi.PiAsyncClient
com.amazonaws.services.pi.AWSPIClient	software.amazon.awssdk.services.pi.PiClient
com.amazonaws.services.pinpoint.AmazonPinpointAsyncClient	software.amazon.awssdk.services.pinpoint.PinpointAsyncClient
com.amazonaws.services.pinpoint.AmazonPinpointClient	software.amazon.awssdk.services.pinpoint.PinpointClient
com.amazonaws.services.polly.AmazonPollyAsyncClient	software.amazon.awssdk.services.polly.PollyAsyncClient
com.amazonaws.services.polly.AmazonPollyClient	software.amazon.awssdk.services.polly.PollyClient
com.amazonaws.services.pricing.AWSPricingAsyncClient	software.amazon.awssdk.services.pricing.PricingAsyncClient

1.x Client	2.x Client
com.amazonaws.services.pricing.AWSPricingClient	software.amazon.awssdk.services.pricing.PricingClient
com.amazonaws.services.rds.AmazonRDSAsyncClient	software.amazon.awssdk.services.rds.RdsAsyncClient
com.amazonaws.services.rds.AmazonRDSClient	software.amazon.awssdk.services.rds.RdsClient
com.amazonaws.services.redshift.AmazonRedshiftAsyncClient	software.amazon.awssdk.services.redshift.RedshiftAsyncClient
com.amazonaws.services.redshift.AmazonRedshiftClient	software.amazon.awssdk.services.redshift.RedshiftClient
com.amazonaws.services.rekognition.AmazonRekognitionAsyncClient	software.amazon.awssdk.services.rekognition.RekognitionAsyncClient
com.amazonaws.services.rekognition.AmazonRekognitionClient	software.amazon.awssdk.services.rekognition.RekognitionClient
com.amazonaws.services.resourcegroups.AWSResourceGroupsAsyncClient	software.amazon.awssdk.services.resourcegroups.ResourceGroupsAsyncClient
com.amazonaws.services.resourcegroups.AWSResourceGroupsClient	software.amazon.awssdk.services.resourcegroups.ResourceGroupsClient
com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIAsyncClient	software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiAsyncClient

1.x Client	2.x Client
com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIClient	software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiClient
com.amazonaws.services.route53.AmazonRoute53AsyncClient	software.amazon.awssdk.services.route53.Route53AsyncClient
com.amazonaws.services.route53.AmazonRoute53Client	software.amazon.awssdk.services.route53.Route53Client
com.amazonaws.services.route53domains.AmazonRoute53DomainsAsyncClient	software.amazon.awssdk.services.route53domains.Route53DomainsAsyncClient
com.amazonaws.services.route53domains.AmazonRoute53DomainsClient	software.amazon.awssdk.services.route53domains.Route53DomainsClient
com.amazonaws.services.s3.AmazonS3Client	software.amazon.awssdk.services.s3.S3Client
com.amazonaws.services.sagemaker.AmazonSageMakerAsyncClient	software.amazon.awssdk.services.sagemaker.SageMakerAsyncClient
com.amazonaws.services.sagemaker.AmazonSageMakerClient	software.amazon.awssdk.services.sagemaker.SageMakerClient
com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeAsyncClient	software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeAsyncClient
com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeClient	software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeClient

1.x Client	2.x Client
com.amazonaws.services.secretsmanager.AWSSecretsManagerAsyncClient	software.amazon.awssdk.services.secretsmanager.SecretsManagerAsyncClient
com.amazonaws.services.secretsmanager.AWSSecretsManagerClient	software.amazon.awssdk.services.secretsmanager.SecretsManagerClient
com.amazonaws.services.securitytoken.AWSSecurityTokenServiceAsyncClient	software.amazon.awssdk.services.sts.StsAsyncClient
com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient	software.amazon.awssdk.services.sts.StsClient
com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryAsyncClient	software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryAsyncClient
com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryClient	software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryClient
com.amazonaws.services.servermigration.AWSServerMigrationAsyncClient	software.amazon.awssdk.services.sms.SmsAsyncClient
com.amazonaws.services.servermigration.AWSServerMigrationClient	software.amazon.awssdk.services.sms.SmsClient

1.x Client	2.x Client
com.amazonaws.services.servicecatalog.AWSServiceCatalogAsyncClient	software.amazon.awssdk.services.servicecatalog.ServiceCatalogAsyncClient
com.amazonaws.services.servicecatalog.AWSServiceCatalogClient	software.amazon.awssdk.services.servicecatalog.ServiceCatalogClient
com.amazonaws.services.servicediscovery.AWSServiceDiscoveryAsyncClient	software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryAsyncClient
com.amazonaws.services.servicediscovery.AWSServiceDiscoveryClient	software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryClient
com.amazonaws.services.shield.AWSShieldAsyncClient	software.amazon.awssdk.services.shield.ShieldAsyncClient
com.amazonaws.services.shield.AWSShieldClient	software.amazon.awssdk.services.shield.ShieldClient
com.amazonaws.services.simpledb.AmazonSimpleDBAsyncClient	software.amazon.awssdk.services.simpledb.SimpleDbAsyncClient
com.amazonaws.services.simpledb.AmazonSimpleDBClient	software.amazon.awssdk.services.simpledb.SimpleDbClient
com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceAsyncClient	software.amazon.awssdk.services.ses.SesAsyncClient
com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient	software.amazon.awssdk.services.ses.SesClient

1.x Client	2.x Client
com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementAsyncClient	software.amazon.awssdk.services.ssm.SsmAsyncClient
com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementClient	software.amazon.awssdk.services.ssm.SsmClient
com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowAsyncClient	software.amazon.awssdk.services.swf.SwfAsyncClient
com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClient	software.amazon.awssdk.services.swf.SwfClient
com.amazonaws.services.snowball.AmazonSnowballAsyncClient	software.amazon.awssdk.services.snowball.SnowballAsyncClient
com.amazonaws.services.snowball.AmazonSnowballClient	software.amazon.awssdk.services.snowball.SnowballClient
com.amazonaws.services.sns.AmazonSNSAsyncClient	software.amazon.awssdk.services.sns.SnsAsyncClient
com.amazonaws.services.sns.AmazonSNSClient	software.amazon.awssdk.services.sns.SnsClient
com.amazonaws.services.sqs.AmazonSQSAsyncClient	software.amazon.awssdk.services.sqs.SqsAsyncClient
com.amazonaws.services.sqs.AmazonSQSClient	software.amazon.awssdk.services.sqs.SqsClient

1.x Client	2.x Client
com.amazonaws.services.stepfunctions.AWSStepFunctionsAsyncClient	software.amazon.awssdk.services.sfn.SfnAsyncClient
com.amazonaws.services.stepfunctions.AWSStepFunctionsClient	software.amazon.awssdk.services.sfn.SfnClient
com.amazonaws.services.storagegateway.AWSStorageGatewayAsyncClient	software.amazon.awssdk.services.storagegateway.StorageGatewayAsyncClient
com.amazonaws.services.storagegateway.AWSStorageGatewayClient	software.amazon.awssdk.services.storagegateway.StorageGatewayClient
com.amazonaws.services.support.AWSSupportAsyncClient	software.amazon.awssdk.services.support.SupportAsyncClient
com.amazonaws.services.support.AWSSupportClient	software.amazon.awssdk.services.support.SupportClient
com.amazonaws.services.transcribe.AmazonTranscribeAsyncClient	software.amazon.awssdk.services.transcribe.TranscribeAsyncClient
com.amazonaws.services.transcribe.AmazonTranscribeClient	software.amazon.awssdk.services.transcribe.TranscribeClient
com.amazonaws.services.translate.AmazonTranslateAsyncClient	software.amazon.awssdk.services.translate.TranslateAsyncClient
com.amazonaws.services.translate.AmazonTranslateClient	software.amazon.awssdk.services.translate.TranslateClient

1.x Client	2.x Client
com.amazonaws.services.waf.AWSWAFAsyncClient	software.amazon.awssdk.services.waf.WafAsyncClient
com.amazonaws.services.waf.AWSWAFClient	software.amazon.awssdk.services.waf.WafClient
com.amazonaws.services.waf.AWSWAFRegionalAsyncClient	software.amazon.awssdk.services.waf.regional.WafRegionalAsyncClient
com.amazonaws.services.waf.AWSWAFRegionalClient	software.amazon.awssdk.services.waf.regional.WafRegionalClient
com.amazonaws.services.workdocs.AmazonWorkDocsAsyncClient	software.amazon.awssdk.services.workdocs.WorkDocsAsyncClient
com.amazonaws.services.workdocs.AmazonWorkDocsClient	software.amazon.awssdk.services.workdocs.WorkDocsClient
com.amazonaws.services.workmail.AmazonWorkMailAsyncClient	software.amazon.awssdk.services.workmail.WorkMailAsyncClient
com.amazonaws.services.workmail.AmazonWorkMailClient	software.amazon.awssdk.services.workmail.WorkMailClient
com.amazonaws.services.workspaces.AmazonWorkspacesAsyncClient	software.amazon.awssdk.services.workspaces.WorkSpacesAsyncClient
com.amazonaws.services.workspaces.AmazonWorkspacesClient	software.amazon.awssdk.services.workspaces.WorkSpacesClient

1.x Client	2.x Client
com.amazonaws.services.xray .AWSXRayAsyncClient	software.amazon.awssdk.serv ices.xray.XRayAsyncClient
com.amazonaws.services.xray .AWSXRayClient	software.amazon.awssdk.serv ices.xray.XRayClient

Client creation defaults

In version 2.x, the following changes have been made to the default client creation logic.

- The default credential provider chain for S3 no longer includes anonymous credentials. You must manually specify anonymous access to S3 by using the `AnonymousCredentialsProvider`.
- The following environment variables related to default client creation are different.

1.x	2.x
AWS_CBOR_DISABLED	CBOR_ENABLED
AWS_ION_BINARY_DISABLE	BINARY_ION_ENABLED

- The following system properties related to default client creation are different.

1.x	2.x
com.amazonaws.sdk.disableEc 2Metadata	aws.disableEc2Metadata
com.amazonaws.sdk.ec2Metada taServiceEndpointOverride	aws.ec2MetadataServiceEndpoint
com.amazonaws.sdk.disableCbor	aws.cborEnabled
com.amazonaws.sdk.disableIo nBinary	aws.binaryIonEnabled

- Version 2.x does not support the following system properties.

1.x
`com.amazonaws.sdk.disableCertChecking`
`com.amazonaws.sdk.enableDefaultMetrics`
`com.amazonaws.sdk.enableThrottledRetry`
`com.amazonaws.regions.RegionUtils.fileOverride`
`com.amazonaws.regions.RegionUtils.disableRemote`
`com.amazonaws.services.s3.disableImplicitGlobalClients`
`com.amazonaws.sdk.enableInRegionOptimizedMode`
- Loading Region configuration from a custom `endpoints.json` file is no longer supported.

Client configuration

In 1.x, SDK client configuration was modified by setting a `ClientConfiguration` instance on the client or client builder. In version 2.x, the client configuration is split into separate configuration classes. With the separate configuration classes, you can configure different HTTP clients for async versus synchronous clients but still use the same `ClientOverrideConfiguration` class.

Example of client configuration in version 1.x

```
AmazonDynamoDBClientBuilder.standard()
.withClientConfiguration(clientConfiguration)
.build()
```

Example of synchronous client configuration in version 2.x

```
ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder();

ApacheHttpClient.Builder httpClientBuilder =
        ApacheHttpClient.builder()
                        .proxyConfiguration(proxyConfig.build());

ClientOverrideConfiguration.Builder overrideConfig =
```

```
        ClientOverrideConfiguration.builder();

DynamoDbClient client =
        DynamoDbClient.builder()
                      .httpClientBuilder(httpClientBuilder)
                      .overrideConfiguration(overrideConfig.build())
                      .build();
```

Example of asynchronous client configuration in version 2.x

```
NettyNioAsyncHttpClient.Builder httpClientBuilder =
        NettyNioAsyncHttpClient.builder();

ClientOverrideConfiguration.Builder overrideConfig =
        ClientOverrideConfiguration.builder();

ClientAsyncConfiguration.Builder asyncConfig =
        ClientAsyncConfiguration.builder();

DynamoDbAsyncClient client =
        DynamoDbAsyncClient.builder()
                           .httpClientBuilder(httpClientBuilder)
                           .overrideConfiguration(overrideConfig.build())
                           .asyncConfiguration(asyncConfig.build())
                           .build();
```

HTTP clients

Notable changes

- In version 2.x, you can change which HTTP client to use at runtime by specifying an implementation using `clientBuilder.httpClientBuilder`.
- When you pass an HTTP client by using `clientBuilder.httpClient` to a service client builder, the HTTP client is not closed by default if the service client closes. This allows you to share HTTP clients between service clients.
- Asynchronous HTTP clients now use non-blocking IO.
- Some operations now use HTTP/2 for improved performance.

Settings changes

Setting	1.x	2.x Sync, Apache	2.x Async, Netty
	`ClientConfiguration clientConfig = new ClientConfiguration()`	`ApacheHttpClient.Builder httpClientBuilder = ApacheHttpClient.builder()`	`NettyNioAsyncHttpClient.Builder httpClientBuilder = NettyNioAsyncHttpClient.builder()`
Max connections	`clientConfig.setMaxConnections(...)` `clientConfig.withMaxConnections(...)`	`httpClientBuilder.maxConnections(...)`	`httpClientBuilder.maxConcurrency(...)`
Connection timeout	`clientConfig.setConnectionTimeout(...)` `clientConfig.withConnectionTimeout(...)`	`httpClientBuilder.connectionTimeout(...)` `httpClientBuilder.connectionAcquisitionTimeout(...)`	`httpClientBuilder.connectionTimeout(...)`
Socket timeout	`clientConfig.setSocketTimeout(...)` `clientConfig.withSo`	`httpClientBuilder.socketTimeout(...)`	`httpClientBuilder.writeTimeout(...)` `httpClientBuilder.`

Setting	1.x	2.x Sync, Apache	2.x Async, Netty
	`cketTimeout(...)`		`readTimeout(...)`
Connection TTL	`clientConfig.setConnectionTTL(...)` `clientConfig.withConnectionTTL(...)`	`httpClientBuilder.connectionTimeToLive(...)`	`httpClientBuilder.connectionTimeToLive(...)`
Connection max idle	`clientConfig.setConnectionMaxIdleMillis(...)` `clientConfig.withConnectionMaxIdleMillis(...)`	`httpClientBuilder.connectionMaxIdleTime(...)`	`httpClientBuilder.connectionMaxIdleTime(...)`
Validate after inactivity	`clientConfig.setValidateAfterInactivityMillis(...)` `clientConfig.withValidateAfterInactivityMillis(...)`	Not supported (Request Feature)	Not supported (Request Feature)

Setting	1.x	2.x Sync, Apache	2.x Async, Netty
Local address	`clientConfig.setLocalAddress(...)` `clientConfig.withLocalAddress(...)`	`httpClientBuilder.localAddress(...)`	Not supported
Expect-continue enabled	`clientConfig.setUseExpectContinue(...)` `clientConfig.withUseExpectContinue(...)`	`httpClientBuilder.expectContinueEnabled(...)`	Not supported (Request Feature)
Connection reaper	`clientConfig.setUseReaper(...)` `clientConfig.withReaper(...)`	`httpClientBuilder.useIdleConnectionReaper(...)`	`httpClientBuilder.useIdleConnectionReaper(...)`
	`AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()`	`DynamoDbClient.builder() .httpClientBuilder(httpClientBuilder) .build()`	`DynamoDbAsyncClient.builder() .httpClientBuilder(httpClientBuilder) .build()`

HTTP client proxies

Settings	1.x	2.x Sync, Apache	2.x Async, Netty
	`ClientConfiguration clientConfig = new ClientConfiguration()`	`ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder()`	`ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder()`
Proxy host	`clientConfig.setProxyHost(...)` `clientConfig.withProxyHost(...)`	`proxyConfig.endpoint(...)`	`proxyConfig.host(...)`
Proxy port	`clientConfig.setProxyPort(...)` `clientConfig.withProxyPort(...)`	`proxyConfig.endpoint(...)` Proxy port is embedded in `endpoint`	`proxyConfig.port(...)`
Proxy username	`clientConfig.setProxyUsername(...)` `clientConfig.withProxyUsername(...)`	`proxyConfig.username(...)`	`proxyConfig.username(...)`

Settings	1.x	2.x Sync, Apache	2.x Async, Netty
Proxy password	`clientConfig.setProxyPassword(...)` `clientConfig.withProxyPassword(...)`	`proxyConfig.password(...)`	`proxyConfig.password(...)`
Proxy domain	`clientConfig.setProxyDomain(...)` `clientConfig.withProxyDomain(...)`	`proxyConfig.ntlmDomain(...)`	Not Supported (Request Feature)
Proxy workstation	`clientConfig.setProxyWorkspace(...)` `clientConfig.withProxyWorkstation(...)`	`proxyConfig.ntlmWorkstation(...)`	Not Supported (Request Feature)
Proxy authentication methods	`clientConfig.setProxyAuthenticationMethods(...)` `clientConfig.withProxyAuthenticationMethods(...)`	Not Supported	Not Supported (Request Feature)

Settings	1.x	2.x Sync, Apache	2.x Async, Netty
Preemptive basic proxy authentication	`clientConfig.setPreemptiveBasicProxyAuth(...)` `clientConfig.withPreemptiveBasicProxyAuth(...)`	`proxyConfig.preemptiveBasicAuthenticationEnabled(...)`	Not Supported (Request Feature)
Non-proxy hosts	`clientConfig.setNonProxyHosts(...)` `clientConfig.withNonProxyHosts(...)`	`proxyConfig.nonProxyHosts(...)`	`proxyConfig.nonProxyHosts(...)`
Disable socket proxy	`clientConfig.setDisableSocketProxy(...)` `clientConfig.withDisableSocketProxy(...)`	Not Supported (Request Feature)	Not Supported (Request Feature)

Settings	1.x	2.x Sync, Apache	2.x Async, Netty
	`AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()`	`httpClientBuilder.proxyConfiguration( proxyConfig.build())`	`httpClientBuilder.proxyConfiguration( proxyConfig.build())`

Client overrides

Setting	1.x	2.x
	`ClientConfiguration clientConfig = new ClientConfiguration()`	`ClientOverrideConfiguration.Builder overrideConfig = ClientOverrideConfiguration.builder()`
User agent prefix	`clientConfig.setUserAgentPrefix(...)` `clientConfig.withUserAgentPrefix(...)`	`overrideConfig.advancedOption( SdkAdvancedClientOption.USER_AGENT_PREFIX, ...)`
User agent suffix	`clientConfig.setUserAgentSuffix(...)` `clientConfig.withUserAgentSuffix(...)`	`overrideConfig.advancedOption( SdkAdvancedClientOption.USER_AGENT_SUFFIX, ...)`
Signer	`clientConfig.setSignerOverride(...)`	`overrideConfig.advancedOption(`

Setting	1.x	2.x
	`clientConfig.withSignerOverride(...)`	`SdkAdvancedClientOption.SIGNER, ...)`
Additional headers	`clientConfig.addHeader(...)` `clientConfig.withHeader(...)`	`overrideConfig.putHeader(...)`
Request timeout	`clientConfig.setRequestTimeout(...)` `clientConfig.withRequestTimeout(...)`	`overrideConfig.apiCallAttemptTimeout(...)`
Client execution timeout	`clientConfig.setClientExecutionTimeout(...)` `clientConfig.withClientExecutionTimeout(...)`	`overrideConfig.apiCallTimeout(...)`
Use Gzip	`clientConfig.setUseGzip(...)` `clientConfig.withGzip(...)`	Not Supported (Request Feature)
Socket buffer size hint	`clientConfig.setSocketBufferSizeHints(...)` `clientConfig.withSocketBufferSizeHints(...)`	Not Supported (Request Feature)

Setting	1.x	2.x
Cache response metadata	`clientConfig.setCacheResponseMetadata(...)` `clientConfig.withCacheResponseMetadata(...)`	Not Supported (Request Feature)
Response metadata cache size	`clientConfig.setResponseMetadataCacheSize(...)` `clientConfig.withResponseMetadataCacheSize(...)`	Not Supported (Request Feature)
DNS resolver	`clientConfig.setDnsResolver(...)` `clientConfig.withDnsResolver(...)`	Not Supported (Request Feature)
TCP keepalive	`clientConfig.setUseTcpKeepAlive(...)` `clientConfig.withTcpKeepAlive(...)`	This option is now in the HTTP Client configuration `- ApacheHttpClient.builder().tcpKeepAlive(true)` `- NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)`
Secure random	`clientConfig.setSecureRandom(...)` `clientConfig.withSecureRandom(...)`	Not Supported (Request Feature)

Setting	1.x	2.x
	`AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()`	`DynamoDbClient.builder() .httpClientBuilder(httpClientBuilder) .build()`

Client override retries

Setting	1.x	2.x
	`ClientConfiguration clientConfig = new ClientConfiguration()`	`RetryPolicy.Builder retryPolicy = RetryPolicy.builder()`
Max error retry	`clientConfig.setMaxErrorRetry(...)` `clientConfig.withMaxErrorRetry(...)`	`retryPolicy.numRetries(...)`
Use throttled retries	`clientConfig.setUseThrottleRetries(...)` `clientConfig.withUseThrottleRetries(...)`	Not supported
Max consecutive retries before throttling	`clientConfig.setMaxConsecutiveRetriesBeforeThrottling(...)` `clientConfig.withMaxConsecutiveRetriesBeforeThrottling(...)`	Not supported

Setting	1.x	2.x

	`AmazonDynamoDBClientBuilder.standard()`
	`    .withClientConfiguration(clientConfiguration)`
	`    .build()`
		`DynamoDbClient.builder()`
		`    .httpClientBuilder(httpClientBuilder)`
		`    .build()`

Asynchronous clients

Setting	1.x	2.x
		`ClientAsyncConfiguration.Builder asyncConfig =` `    ClientAsyncConfiguration.builder()`

Executor	`AmazonDynamoDBAsyncClientBuilder.standard()`
	`    .withExecutorFactory(...)`
	`    .build()`
		`asyncConfig.advancedOption(`
		`    SdkAdvancedAsyncClientOption.FUTURE_COMPLETION_EXECUTOR, ...)`

		`DynamoDbAsyncClient.builder()`
		`    .asyncConfiguration(asyncConfig)`
		`    .build()`

Other client changes

The following `ClientConfiguration` option from 1.x has changed in 2.x of the SDK and doesn't have a direct equivalent.

Setting	1.x	2.x equivalent
Protocol	`clientConfig.setProtocol(Protocol.HTTP)` `clientConfig.withProtocol(Protocol.HTTP)`	The protocol setting is HTTPS by default. To modify the setting, specify the protocol setting an HTTP endpoint on the client builder: `clientBuilder.endpointOverride( URI.create("http://..."))`

Credentials provider changes

This section provides a mapping of the name changes of credentials provider classes and methods between versions 1.x and 2.x of the AWS SDK for Java.

Notable differences

- The default credentials provider loads system properties before environment variables in version 2.x. For more information, see Using credentials.
- The constructor method is replaced with the `create` or `builder` methods.

  Example

  ```
  DefaultCredentialsProvider.create();
  ```

- Asynchronous refresh is no longer set by default. You must specify it with the `builder` of the credentials provider.

  Example

  ```
  ContainerCredentialsProvider provider = ContainerCredentialsProvider.builder()
            .asyncCredentialUpdateEnabled(true)
            .build();
  ```

- You can specify a path to a custom profile file using the `ProfileCredentialsProvider.builder()`.

  Example

  ```
  ProfileCredentialsProvider profile = ProfileCredentialsProvider.builder()

   .profileFile(ProfileFile.builder().content(Paths.get("myProfileFile.file")).build())
           .build();
  ```

- Profile file format has changed to more closely match the AWS CLI. For details, see [Configuring the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

Credentials provider changes mapped between versions 1.x and 2.x

`AWSCredentialsProvider`

Change category	1.x	2.x
Package/class name	`com.amazonaws.auth.AWSCredentialsProvider`	`software.amazon.awssdk.auth.credentials.AwsCredentialsProvider`
Method name	`getCredentials`	`resolveCredentials`
Unsupported method	`refresh`	Not supported

`DefaultAWSCredentialsProviderChain`

Change category	1.x	2.x
Package/class name	`com.amazonaws.auth.DefaultAWSCredentialsProviderChain`	`software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider`

Change category	1.x	2.x
Creation	new DefaultAWSCredentialsProviderChain	DefaultCredentialsProvider.create
Unsupported method	getInstance	Not supported
Priority order of external settings	Environment variables before system properties	System properties before environment variables

AWSStaticCredentialsProvider

Change category	1.x	2.x
Package/class name	com.amazonaws.auth.AWSStaticCredentialsProvider	software.amazon.awssdk.auth.credentials.StaticCredentialsProvider
Creation	new AWSStaticCredentialsProvider	StaticCredentialsProvider.create

EnvironmentVariableCredentialsProvider

Change category	1.x	2.x
Package/class name	com.amazonaws.auth.EnvironmentVariableCredentialsProvider	software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider
Creation	new EnvironmentVariableCredentialsProvider	EnvironmentVariableCredentialsProvider.create

Change category	1.x	2.x
Environment variable name	`AWS_ACCESS_KEY`	`AWS_ACCESS_KEY_ID`
	`AWS_SECRET_KEY`	`AWS_SECRET_ACCESS_KEY`

`SystemPropertiesCredentialsProvider`

Change category	1.x	2.x
Package/class name	`com.amazonaws.auth.SystemPropertiesCredentialsProvider`	`software.amazon.awssdk.auth.credentials.SystemPropertyCredentialsProvider`
Creation	`new SystemPropertiesCredentialsProvider`	`SystemPropertiesCredentialsProvider.create`
System property name	`aws.secretKey`	`aws.secretAccessKey`

`ProfileCredentialsProvider`

Change category	1.x	2.x
Package/class name	`com.amazonaws.auth.profile.ProfileCredentialsProvider`	`software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider`
Creation	`new ProfileCredentialsProvider`	`ProfileCredentialsProvider.create`
Location of custom profile	• `AWS_CREDENTIAL_PROFILES_FILE` environment variable	• `AWS_SHARED_CREDENTIALS_FILE` environment variable

Change category	1.x	2.x
	• `new ProfileCredentialsProvider`	• `ProfileCredentialsProvider.builder`

`ContainerCredentialsProvider`

Change category	1.x	2.x
Package/class name	`com.amazonaws.auth.ContainerCredentialsProvider`	`software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider`
Creation	`new ContainerCredentialsProvider`	`ContainerCredentialsProvider.create`
Specify asynchronous refresh	Not supported	Default behavior

`InstanceProfileCredentialsProvider`

Change category	1.x	2.x
Package/class name	`com.amazonaws.auth.InstanceProfileCredentialsProvider`	`software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider`
Creation	`new InstanceProfileCredentialsProvider`	`InstanceProfileCredentialsProvider.create`
Specify asynchronous refresh	`new InstanceProfileCredentialsProvider(true)`	`InstanceProfilerCredentialProvider.builder().asyncCred`

Change category	1.x	2.x
		`entialUpdateEnabled(true).build()`
System property name	`com.amazonaws.sdk.disableEc2Metadata`	`aws.disableEc2Metadata`
	`com.amazonaws.sdk.ec2MetadataServiceEndpointOverride`	`aws.ec2MetadataServiceEndpoint`

`STSAssumeRoleSessionCredentialsProvider`

Change category	1.x	2.x
Package/class name	`com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider`	`software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider`
Creation	• `new STSAssumeRoleSessionCredentialsProvider` • `new STSAssumeRoleSessionCredentialsProvider.Builder`	`StsAssumeRoleCredentialsProvider.builder`
Asynchronous refresh	Default behavior	Default behavior
Configuration	`new STSAssumeRoleSessionCredentialsProvider.Builder`	Configure a `StsClient` and `AssumeRoleRequest` request

`STSSessionCredentialsProvider`

Change category	1.x	2.x
Package/class name	`com.amazonaws.auth.STSSessionCredentialsProvider`	`software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider`
Creation	`new STSAssumeRoleSessionCredentialsProvider`	`StsGetSessionTokenCredentialsProvider.builder`
Asynchronous refresh	Default behavior	`StsGetSessionTokenCredentialsProvider.builder`
Configuration	Constructor parameters	Configure an `StsClient` and `GetSessionTokenRequest` request in a builder

`WebIdentityFederationSessionCredentialsProvider`

Change category	1.x	2.x
Package/class name	`com.amazonaws.auth.WebIdentityFederationSessionCredentialsProvider`	`software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProviderr`
Creation	`new WebIdentityFederationSessionCredentialsProvider`	`StsAssumeRoleWithWebIdentityCredentialsProvider.builder`

Change category	1.x	2.x
Asynchronous refresh	Default behavior	`StsAssumeRoleWithWebIdentityCredentialsProvider.builder`
Configuration	Constructor parameters	Configure an `StsClient` and `AssumeRoleWithWebIdentityRequest` request in a builder

Classes replaced

1.x class	2.x replacement classes
`com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper`	`software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider` and `software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider`
`com.amazonaws.services.s3.S3CredentialsProviderChain`	`software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider` and `software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider`

Classes removed

1.x class
`com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider`
`com.amazonaws.auth.PropertiesFileCredentialsProvider`

Region changes

This section describes the changes implemented in the AWS SDK for Java 2.x for using the `Region` and `Regions` classes.

Region configuration

- Some AWS services don't have Region specific endpoints. When using those services, you must set the Region as `Region.AWS_GLOBAL` or `Region.AWS_CN_GLOBAL`.

  Example

  ```
  Region region = Region.AWS_GLOBAL;
  ```

- `com.amazonaws.regions.Regions` and `com.amazonaws.regions.Region` classes are now combined into one class, `software.amazon.awssdk.regions.Region`.

Method and class name mappings

The following tables map Region related classes between versions 1.x and 2.x of the AWS SDK for Java. You can create an instance of these classes using the `of()` method.

Example

```
RegionMetadata regionMetadata = RegionMetadata.of(Region.US_EAST_1);
```

1.x Regions class method changes

1.x	2.x
`Regions.fromName`	`Region.of`
`Regions.getName`	`Region.id`
`Regions.getDescription`	`Region.metadata().description()`
`Regions.getCurrentRegion`	Not Supported
`Regions.DEFAULT_REGION`	Not Supported

1.x	2.x
`Regions.name`	`Region.id`

1.x Region class method changes

1.x	2.x
`Region.getName`	`Region.id`
`Region.hasHttpsEndpoint`	Not Supported
`Region.hasHttpEndpoint`	Not Supported
`Region.getAvailableEndpoints`	Not Supported
`Region.createClient`	Not Supported

RegionMetadata class method changes

1.x	2.x
`RegionMetadata.getName`	`RegionMetadata.name`
`RegionMetadata.getDomain`	`RegionMetadata.domain`
`RegionMetadata.getPartition`	`RegionMetadata.partition`

ServiceMetadata class method changes

1.x	2.x
`Region.getServiceEndpoint`	`ServiceMetadata.endpointFor(Region)`
`Region.isServiceSupported`	`ServiceMetadata.regions().contains(Region)`

Operations, requests and responses changes

In v2.x of the SDK for Java, requests are passed to a client operation. For example `DynamoDbClient's` `PutItemRequest` is passed to `DynamoDbClient.putItem` operation. These operations return a response from the AWS service, such as a `PutItemResponse`.

Version 2.x of the SDK for Java has the following changes from 1.x.

- Operations with multiple response pages now have a `Paginator` method for automatically iterating over all items in the response.
- You cannot mutate requests and responses.
- You must create requests and responses with a static builder method instead of a constructor. For example, 1.x's `new PutItemRequest().withTableName(...)` is now `PutItemRequest.builder().tableName(...).build()`.
- Operations support a short-hand way to create requests: `dynamoDbClient.putItem(request -> request.tableName(...))`.

Streaming operations

Streaming operations such as Amazon S3 `getObject` and `putObject` methods now support non-blocking I/O. As a result, the request and response POJOs no longer take an `InputStream` as a parameter. Instead, for synchronous requests the request object accepts `RequestBody`, which is a stream of bytes. The asynchronous equivalent accepts an `AsyncRequestBody`.

Example of Amazon S3 `putObject` operation in 1.x

```
s3client.putObject(BUCKET, KEY, new File(file_path));
```

Example of Amazon S3 `putObject` operation in 2.x

```
s3client.putObject(PutObjectRequest.builder()
                                  .bucket(BUCKET)
                                  .key(KEY)
                                  .build(),
                 RequestBody.of(Paths.get("myfile.in")));
```

In parallel, a streaming response object accepts a `ResponseTransformer` for synchronous clients and a `AsyncResponseTransformer` for asynchronous clients.

Example of Amazon S3 `getObject` operation in 1.x

```
S3Object o = s3.getObject(bucket, key);
S3ObjectInputStream s3is = o.getObjectContent();
FileOutputStream fos = new FileOutputStream(new File(key));
```

Example of Amazon S3 `getObject` operation in 2.x

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),
  ResponseTransformer.toFile(Paths.get("key")));
```

In the SDK for Java 2.x , streaming response operations have an `AsBytes` method to load the response into memory and simplify common in-memory type conversions.

Exception changes

Exception class names, their structures, and their relationships have changed. `software.amazon.awssdk.core.exception.SdkException` is the new base `Exception` class that all the other exceptions extend.

This table maps the exception class name changes.

1.x	2.x
`com.amazonaws.SdkBaseException` `com.amazonaws.AmazonClientException`	`software.amazon.awssdk.core.exception.SdkException`
`com.amazonaws.SdkClientException`	`software.amazon.awssdk.core.exception.SdkClientException`
`com.amazonaws.AmazonServiceException`	`software.amazon.awssdk.awscore.exception.AwsServiceException`

The following table maps the methods on exception classes between version 1.x and 2.x.

1.x	2.x
AmazonServiceException.getRequestId	SdkServiceException.requestId
AmazonServiceException.getServiceName	AwsServiceException.awsErrorDetails().serviceName
AmazonServiceException.getErrorCode	AwsServiceException.awsErrorDetails().errorCode
AmazonServiceException.getErrorMessage	AwsServiceException.awsErrorDetails().errorMessage
AmazonServiceException.getStatusCode	AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode
AmazonServiceException.getHttpHeaders	AwsServiceException.awsErrorDetails().sdkHttpResponse().headers
AmazonServiceException.rawResponse	AwsServiceException.awsErrorDetails().rawResponse

Serialization changes

The SDK for Java v1.x and v2.x differ in how they serialize List objects to request parameters.

The SDK for Java 1.x does not serialize an empty list, whereas the SDK for Java 2.x serializes an empty list as an empty parameter.

For example, consider a service with a `SampleOperation` that takes a `SampleRequest`. The `SampleRequest` accepts two parameters—a String type `str1` and List type `listParam`—as shown in the following examples.

Example of `SampleOperation` in 1.x

```
SampleRequest v1Request = new SampleRequest()
```

```
    .withStr1("TestName");

sampleServiceV1Client.sampleOperation(v1Request);
```

Wire-level logging shows that the `listParam` parameter is not serialized.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName
```

Example of `SampleOperation` in 2.x

```
sampleServiceV2Client.sampleOperation(b -> b
    .str1("TestName"));
```

Wire-level logging shows that the `listParam` parameter is serialized with no value.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName&listParam=
```

Service-specific changes

Amazon S3 changes

SDK for Java 2.x disables anonymous access by default. As a result, you must enable anonymous access by using the `AnonymousCredentialsProvider`.

Operation name changes

Many of the operation names for the Amazon S3 client have changed in the AWS SDK for Java 2.x. In version 1.x, the Amazon S3 client is not generated directly from the service API. This results in inconsistency between the SDK operations and the service API. In version 2.x, we now generate the Amazon S3 client to be more consistent with the service API.

The following table shows the operation names in the two versions.

Amazon S3 Operation names

1.x	2.x
`abortMultipartUpload`	`abortMultipartUpload`
`changeObjectStorageClass`	`copyObject`

1.x	2.x
completeMultipartUpload	completeMultipartUpload
copyObject	copyObject
copyPart	uploadPartCopy
createBucket	createBucket
deleteBucket	deleteBucket
deleteBucketAnalyticsConfiguration	deleteBucketAnalyticsConfiguration
deleteBucketCrossOriginConfiguration	deleteBucketCors
deleteBucketEncryption	deleteBucketEncryption
deleteBucketInventoryConfiguration	deleteBucketInventoryConfiguration
deleteBucketLifecycleConfiguration	deleteBucketLifecycle
deleteBucketMetricsConfiguration	deleteBucketMetricsConfiguration
deleteBucketPolicy	deleteBucketPolicy
deleteBucketReplicationConfiguration	deleteBucketReplication
deleteBucketTaggingConfiguration	deleteBucketTagging
deleteBucketWebsiteConfiguration	deleteBucketWebsite
deleteObject	deleteObject
deleteObjectTagging	deleteObjectTagging

1.x	2.x
deleteObjects	deleteObjects
deleteVersion	deleteObject
disableRequesterPays	putBucketRequestPayment
doesBucketExist	headBucket
doesBucketExistV2	headBucket
doesObjectExist	headObject
enableRequesterPays	putBucketRequestPayment
generatePresignedUrl	S3Presigner
getBucketAccelerateConfiguration	getBucketAccelerateConfiguration
getBucketAcl	getBucketAcl
getBucketAnalyticsConfiguration	getBucketAnalyticsConfiguration
getBucketCrossOriginConfiguration	getBucketCors
getBucketEncryption	getBucketEncryption
getBucketInventoryConfiguration	getBucketInventoryConfiguration
getBucketLifecycleConfiguration	getBucketLifecycle or getBucketLifecycleConfiguration
getBucketLocation	getBucketLocation
getBucketLoggingConfiguration	getBucketLogging
getBucketMetricsConfiguration	getBucketMetricsConfiguration

1.x	2.x
getBucketNotificationConfiguration	getBucketNotification or getBucketNotificationConfiguration
getBucketPolicy	getBucketPolicy
getBucketReplicationConfiguration	getBucketReplication
getBucketTaggingConfiguration	getBucketTagging
getBucketVersioningConfiguration	getBucketVersioning
getBucketWebsiteConfiguration	getBucketWebsite
getObject	getObject
getObjectAcl	getObjectAcl
getObjectAsString	getObjectAsBytes().asUtf8String
getObjectMetadata	headObject
getObjectTagging	getObjectTagging
getResourceUrl	S3Utilities#getUrl
getS3AccountOwner	listBuckets
getUrl	S3Utilities#getUrl
headBucket	headBucket
initiateMultipartUpload	createMultipartUpload
isRequesterPaysEnabled	getBucketRequestPayment
listBucketAnalyticsConfigurations	listBucketAnalyticsConfigurations

1.x	2.x
listBucketInventoryConfigurations	listBucketInventoryConfigurations
listBucketMetricsConfigurations	listBucketMetricsConfigurations
listBuckets	listBuckets
listMultipartUploads	listMultipartUploads
listNextBatchOfObjects	listObjectsV2Paginator
listNextBatchOfVersions	listObjectVersionsPaginator
listObjects	listObjects
listObjectsV2	listObjectsV2
listParts	listParts
listVersions	listObjectVersions
putObject	putObject
restoreObject	restoreObject
restoreObjectV2	restoreObject
selectObjectContent	selectObjectContent
setBucketAccelerateConfiguration	putBucketAccelerateConfiguration
setBucketAcl	putBucketAcl
setBucketAnalyticsConfiguration	putBucketAnalyticsConfiguration
setBucketCrossOriginConfiguration	putBucketCors
setBucketEncryption	putBucketEncryption

1.x	2.x
`setBucketInventoryConfiguration`	`putBucketInventoryConfiguration`
`setBucketLifecycleConfiguration`	`putBucketLifecycle` or `putBucketLifecycleConfiguration`
`setBucketLoggingConfiguration`	`putBucketLogging`
`setBucketMetricsConfiguration`	`putBucketMetricsConfiguration`
`setBucketNotificationConfiguration`	`putBucketNotification` or `putBucketNotificationConfiguration`
`setBucketPolicy`	`putBucketPolicy`
`setBucketReplicationConfiguration`	`putBucketReplication`
`setBucketTaggingConfiguration`	`putBucketTagging`
`setBucketVersioningConfiguration`	`putBucketVersioning`
`setBucketWebsiteConfiguration`	`putBucketWebsite`
`setObjectAcl`	`putObjectAcl`
`setObjectRedirectLocation`	`copyObject`
`setObjectTagging`	`putObjectTagging`
`uploadPart`	`uploadPart`

Amazon SNS changes

An SNS client can no longer access SNS topics in Regions other than the Region that it is configured to access.

Amazon SQS changes

An SQS client can no longer access SQS queues in Regions other than the Region that it is configured to access.

Amazon RDS changes

The SDK for Java 2.x uses `RdsUtilities#generateAuthenticationToken` in place of the class `RdsIamAuthTokenGenerator` in 1.x.

Profile file changes

The AWS SDK for Java 2.x parses the profile definitions in `~/.aws/config` and `~/.aws/credentials` to more closely emulate the way the AWS CLI parses the files.

The SDK for Java 2.x:

- Resolves a `~/` or `~` followed by the file system's default path separator at the start of the path by checking, in order, `$HOME`, `$USERPROFILE` (Windows only), `$HOMEDRIVE`, `$HOMEPATH` (Windows only), and then the `user.home` system property.
- Looks for the `AWS_SHARED_CREDENTIALS_FILE` environment variable instead of `AWS_CREDENTIAL_PROFILES_FILE`.
- Silently drops profile definitions in configuration files without the word `profile` at the beginning of the profile name.
- Silently drops profile definitions that do not consist of alphanumeric, underscore or dash characters (after the leading `profile` word has been removed for configuration files).
- Merges settings of profile definitions duplicated within the same file.
- Merges settings of profile definitions duplicated in both the configuration and credentials files.
- Does NOT merge settings if both `[profile foo]` and `[foo]` are found in the same file.
- Uses settings in `[profile foo]` if both `[profile foo]` and `[foo]` are found in the configuration file.
- Uses the value of the last duplicated setting in the same file and profile.
- Recognizes both `;` and `#` for defining a comment.
- Recognizes `;` and `#` in profile definitions to define a comment, even if the characters are adjacent to the closing bracket.

- Recognizes ; and # to define a comment only in setting values only if they are preceded by whitespace.
- Recognizes ; and # and all following content in setting values if they are not preceded by whitespace.
- Considers role-based credentials the highest-priority credentials. The 2.x SDK always uses role-based credentials if the user specifies the `role_arn` property.
- Considers session-based credentials the second-highest-priority credentials. The 2.x SDK always uses session-based credentials if role-based credentials were not used and the user specifies the `aws_access_key_id` and `aws_session_token` properties.
- Uses basic credentials if role-based and session-based credentials are not used and the user specified the `aws_access_key_id` property.

Environment variables and system properties changes

1.x Environment Variable	1.x System Property	2.x Environment Variable	2.x System Property
`AWS_ACCESS_KEY_ID` `AWS_ACCESS_KEY`	`aws.accessKeyId`	`AWS_ACCESS_KEY_ID`	`aws.accessKeyId`
`AWS_SECRET_KEY` `AWS_SECRET_ACCESS_KEY`	`aws.secretKey`	`AWS_SECRET_ACCESS_KEY`	`aws.secretAccessKey`
`AWS_SESSION_TOKEN`	`aws.sessionToken`	`AWS_SESSION_TOKEN`	`aws.sessionToken`
`AWS_REGION`	`aws.region`	`AWS_REGION`	`aws.region`
`AWS_CONFIG_FILE`		`AWS_CONFIG_FILE`	`aws.configFile`
`AWS_CREDENTIAL_PROFILES_FILE`		`AWS_SHARED_CREDENTIALS_FILE`	`aws.sharedCredentialsFile`

1.x Environment Variable	1.x System Property	2.x Environment Variable	2.x System Property
AWS_PROFILE	aws.profile	AWS_PROFILE	aws.profile
AWS_EC2_METADATA_DISABLED	com.amazonaws.sdk.disableEc2Metadata	AWS_EC2_METADATA_DISABLED	aws.disableEc2Metadata
	com.amazonaws.sdk.ec2MetadataServiceEndpointOverride	AWS_EC2_METADATA_SERVICE_ENDPOINT	aws.ec2MetadataServiceEndpoint
AWS_CONTAINER_CREDENTIALS_RELATIVE_URI		AWS_CONTAINER_CREDENTIALS_RELATIVE_URI	aws.containerCredentialsPath
AWS_CONTAINER_CREDENTIALS_FULL_URI		AWS_CONTAINER_CREDENTIALS_FULL_URI	aws.containerCredentialsFullUri
AWS_CONTAINER_AUTHORIZATION_TOKEN		AWS_CONTAINER_AUTHORIZATION_TOKEN	aws.containerAuthorizationToken
AWS_CBOR_DISABLED	com.amazonaws.sdk.disableCbor	CBOR_ENABLED	aws.cborEnabled

1.x Environment Variable	1.x System Property	2.x Environment Variable	2.x System Property
AWS_ION_BINARY_DISABLE	com.amazonaws.sdk.disableIonBinary	BINARY_ION_ENABLED	aws.binaryIonEnabled
AWS_EXECUTION_ENV		AWS_EXECUTION_ENV	aws.executionEnvironment
	com.amazonaws.sdk.disableCertChecking	Not supported (Request feature)	Not supported (Request feature)
	com.amazonaws.sdk.enableDefaultMetrics	Not supported	Not supported
	com.amazonaws.sdk.enableThrottledRetry	Not supported	Not supported
	com.amazonaws.regions.RegionUtils.fileOverride	Not supported (Request feature)	Not supported (Request feature)
	com.amazonaws.regions.RegionUtils.disableRemote	Not supported (Request feature)	Not supported (Request feature)

1.x Environment Variable	1.x System Property	2.x Environment Variable	2.x System Property
	`com.amazonaws.services.s3.disableImplicitGlobalClients`	Not supported (Request feature)	Not supported (Request feature)
	`com.amazonaws.sdk.enableInRegionOptimizedMode`	Not supported (Request feature)	Not supported (Request feature)

Changes in Waiters from version 1 to version 2

This topic details the changes in the functionality of Waiters from version 1 (v1) to version 2 (v2).

The following tables demonstrate the difference for DynamoDB waiters specifically. Waiters for other services follow the same pattern.

High-level changes

Waiters classes are in the same Maven artifact as the service.

Change	v1	v2

Maven dependencies	`<dependencyManagement>`
	`    <dependencies>`
	`        <dependency>`
	`            <groupId>com.amazonaws</groupId>`
	`            <artifactId>aws-java-sdk-bom</artifactId>`
		`<dependencyManagement>`
		`    <dependencies>`
		`        <dependency>`
		`            <groupId>software.amazon.awssdk</groupId>`
		`            <artifactId>bom</artifactId>`

Change	v1	v2
	<version>1.12.680[1]</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</groupId> <artifactId>dynamodb</artifactId> </dependency> </dependencies>	<version>2.25.10[2]</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>dynamodb</artifactId> </dependency> </dependencies>
Package name	`com.amazonaws.services.dynamodbv2.waiters`	`software.amazon.awssdk.services.dynamodb.waiters`
Class names	AmazonDynamoDBWaiters	• Synchronous: DynamoDbWaiter • Asynchronous: DynamoDbAsyncWaiter

[1] Latest version. [2] Latest version.

API changes

<table>
<tr><th>Change</th><th>v1</th><th>v2</th></tr>
<tr><td>Create a waiter</td><td>

```
AmazonDynamoDB client
 = AmazonDynamoDBClie
ntBuilder

    .standard().build(
);
AmazonDynamoDBWaiters
 waiter = client.wa
iters();
```

</td><td>

Synchronous:

```
DynamoDbClient client
 = DynamoDbClient.cre
ate();
DynamoDbWaiter waiter
 = client.waiter();
```

Asynchronous:

```
DynamoDbAsyncClient
 asyncClient =
        DynamoDbA
syncClient.create();
DynamoDbAsyncWaiter
 waiter = asyncClie
nt.waiter();
```

</td></tr>
<tr><td>Wait until a table exists</td><td>

Synchronous:

```
waiter.tableExists()
    .run(new WaiterPar
ameters<>(
        new DescribeT
ableRequest(tableN
ame)));
```

Asynchronous:

```
waiter.tableExists()
    .runAsync(new
 WaiterParameters()
      .withRequest(new
 DescribeTableReque
st(tableName)),
```

</td><td>

Synchronous:

```
WaiterResponse<Des
cribeTableResponse>
 waiterResponse =
    waiter.waitUntilTa
bleExists(
        r -> r.tableNa
me("myTable"));
waiterResponse.match
ed().response()
       .ifPresen
t(System.out::prin
tln);
```

Asynchronous:

</td></tr>
</table>

<table>
<tr><th>Change</th><th>v1</th><th>v2</th></tr>
<tr><td></td><td><pre><code>   new WaiterHandler() {
              @Override
              public void onWaitSuccess(AmazonWebServiceRequest amazonWebServiceRequest) {
                  System.out.println("Table creation succeeded");
              }

              @Override
              public void onWaitFailure(Exception e) {
                  e.printStackTrace();
              }
          }).get();</code></pre></td><td><pre><code>waiter.waitUntilTableExists(r -> r.tableName(tableName))
           .whenComplete((r, t) -> {
               if (t != null) {
                   t.printStackTrace();
               } else {
                   System.out.println(
                       "Table creation succeeded");
               }
           }).join();</code></pre></td></tr>
</table>

Configuration changes

<table>
<tr><th>Change</th><th>v1</th><th>v2</th></tr>
<tr><td>Polling Strategy (max attempts and fixed delay)</td><td><pre><code>MaxAttemptsRetryStrategy maxAttemptsRetryStrategy =
        new MaxAttemptsRetryStrategy(10);

FixedDelayStrategy fixedDelayStrategy =
        new FixedDelayStrategy(3);</code></pre></td><td><pre><code>FixedDelayBackoffStrategy fixedDelayBackoffStrategy =
        FixedDelayBackoffStrategy
            .create(Duration.ofSeconds(3));</code></pre></td></tr>
</table>

Change	v1	v2
	`PollingStrategy pollingStrategy = new PollingStrategy(maxAttemptsRetryStrategy, fixedDelayStrategy); waiter.tableExists().run( new WaiterParameters<>( new DescribeTableRequest(tableName)), pollingStrategy);`	`waiter.waitUntilTableExists(r -> r.tableName(tableName), c -> c.maxAttempts(10) .backoffStrategy(fixedDelayBackoffStrategy));`

Changes in Amazon S3 Transfer Manager from version 1 to version 2

This topic details the changes in the Amazon S3 Transfer Manager from version 1 (v1) to version 2 (v2).

High-level changes

Change	v1	v2
Maven dependencies	`<dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</groupId> <artifactId>aws-java-sdk-bom</artifactId> <version>1.12.587[1]</version>`	`<dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>bom</artifactId> <version>2.21.21[2]</version> <type>pom</type>`

Change	v1	v2
	<type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</groupId> <artifactId>aws-java-sdk-s3</artifactId> </dependency> </dependencies>	<scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>s3-transfer-manager</artifactId> </dependency> <dependency> <groupId>software.amazon.awssdk.crt</groupId> <artifactId>aws-crt</artifactId> <version>0.28.7[3]</version> </dependency> </dependencies>
Package name	`com.amazonaws.services.s3.transfer`	`software.amazon.awssdk.transfer.s3`
Class name	`TransferManager`	`S3TransferManager`

[1] Latest version. [2] Latest version. [3] Latest version.

Configuration API changes

Setting	v1	v2
(get a builder)	`TransferManagerBuilder tmBuilder = TransferManagerBuilder.standard();`	`S3TransferManager.Builder tmBuilder = S3TransferManager.builder();`
S3 client	`tmBuilder.withS3Client(...);` `tmBuilder.setS3Client(...);`	`tmBuilder.s3Client(...);`
Executor	`tmBuilder.withExecutorFactory(...);` `tmBuilder.setExecutorFactory(...);`	`tmBuilder.executor(...);`
Shutdown thread pools	`tmBuilder.withShutDownThreadPools(...);` `tmBuilder.setShutdownThreadPools(...);`	Not supported. The provided executor will not be shut down when the S3TransferManager is closed

Minimum upload part size	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().minimumPartSizeInBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

Multipart upload threshold	`tmBuilder.withMinimumUploadPartSize(...);`	`S3AsyncClient s3 = S3AsyncClient.crtBuilder().`

Setting	v1	v2

	`tmBuilder.setMinimumUploadPartSize(...);`	`threshold`
		`InBytes(...).build();`
		
		`tmBuilder.s3Client(s3)`
		`;`

Minimum copy part size	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 =`
		`S3AsyncClient.crtBuilder().`
		`minimumPartSizeInBytes(...)`
		`.build();`
		
		`tmBuilder.s3Client(s3);`

Multipart copy threshold	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 =`
		`S3AsyncClient.crtBuilder().`
		`threshold`
		`InBytes(...).build();`
		
		`tmBuilder.s3Client(s3)`
		`;`

Disable parallel downloads	`tmBuilder.withDisableParallelDownloads(...);`
	`tmBuilder.setDisableParallelDownloads(...);`
		Disable parallel downloads by passing a standard Java-based S3 client to the transfer manager.
		
		`S3AsyncClient s3 =`
		`S3AsyncClient.builder().build();`
		
		`tmBuilder.s3Client(s3);`

Setting	v1	v2
Always calculate multipart md5	`tmBuilder.withAlwaysCalculateMultipartMd5(...);` `tmBuilder.setAlwaysCalculateMultipartMd5(...);`	Not supported.

Behavior changes

Parallel transfer requires AWS CRT-based S3 client

In the SDK for Java 2.x, the automatic parallel transfer feature (multipart upload/download) is available through the AWS CRT-based S3 client. To enable the parallel transfer feature , you must explicitly add the AWS Common Runtime (CRT) library dependency for the maximized performance.

The AWS CRT-based S3 client alone—without using `S3TransferManager`—provides maximized performance of parallel transfers. `S3TransferManager` v2 provides additional APIs that make it easier to transfer files and directories.

The ability for the `S3TransferManager` to perform parallel transfers depends on the how `S3TransferManager` is initiated and if the AWS Common Runtime (CRT) library has been declared as a dependency.

The following table describes three initialization scenarios for an `S3TransferManager` v2 with and without the AWS CRT declared as a dependency.

S3TransferManager v2 initialization approach	Is AWS CRT declared as a dependency?	
	yes	no
Initialize the `S3TransferManager` without passing an `S3AsyncClient` instance *Static create method:*	✅	❌

<table>
<tr><th>S3TransferManager v2 initialization approach</th><th colspan="2">Is AWS CRT declared as a dependency?</th></tr>
<tr><td><pre><code>S3TransferManager.create();</code></pre>
- OR -

Builder method:
<pre><code>S3TransferManager.builder().build();</code></pre></td><td>automatic parallel transfer enabled</td><td>automatic parallel transfer disabled</td></tr>
<tr><td>Pass an <code>S3AsyncClient</code> instance that is built with one of the crt*() builder methods
<pre><code>S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder().build();
S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</code></pre>
- OR -
<pre><code>S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</code></pre></td><td>✅

automatic parallel transfer enabled</td><td>❌

runtime error</td></tr>
</table>

S3TransferManager v2 initialization approach	Is AWS CRT declared as a dependency?	

Pass an `S3AsyncClient` instance that is built with one of the standard builder methods so that the transfer manager has no reference to the CRT
`S3AsyncClient s3AsyncClient = S3AsyncClient.builder().build();`
`S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();`
- OR -
`S3AsyncClient s3AsyncClient = S3AsyncClient.create();`
`S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();`
	❌ automatic parallel transfer disabled	❌ automatic parallel transfer disabled

Parallel download via byte-range fetches

When the automatic parallel transfer feature is enabled, the S3 Transfer Manager v2 uses byte-range fetches to retrieve specific portions of the object in parallel (multipart download). The way an object is downloaded with v2 does not depend on how the object was originally uploaded. All downloads can benefit from high throughput and concurrency.

In contrast, with S3 Transfer Manager v1, it does matter how the object was originally uploaded. The S3 Transfer Manager v1 retrieves the parts of the object the same way that the parts were uploaded. If an object was originally uploaded as a single object, the S3 Transfer Manager v1 is not able to accelerate the downloading process by using sub-requests.

Failure behavior

With S3 Transfer Manager v1, a directory transfer request fails if any sub-request fails. Unlike v1, the future returned from S3 Transfer Manager v2 completes successfully even if some sub-requests fail.

As a result, you should check for errors in the response by using the `CompletedDirectoryDownload.failedTransfers()` method or

CompletedDirectoryUpload.failedTransfers() method even when the future completes successfully.

Changes in the EC2 metadata utility from version 1 to version 2

This topic details the changes in the SDK for Java Amazon Elastic Compute Cloud (EC2) metadata utility from version 1 (v1) to version 2 (v2).

High-level changes

Change	v1	v2

Maven dependencies	`<dependencyManagement>`
	`    <dependencies>`
	`        <dependency>`
	`            <groupId>com.amazonaws</groupId>`
	`            <artifactId>aws-java-sdk-bom</artifactId>`
	`            <version>1.12.587[1]</version>`
	`            <type>pom</type>`
	`            <scope>import</scope>`
	`        </dependency>`
	`    </dependencies>`
	`</dependencyManagement>`
	`<dependencies>`
	`    <dependency>`
	`        <groupId>com.amazonaws</groupId>`
	`        <artifactId>aws-java-sdk-core</artifactId>`
	`    </dependency>`
	`</dependencies>`
		`<dependencyManagement>`
		`    <dependencies>`
		`        <dependency>`
		`            <groupId>software.amazon.awssdk</groupId>`
		`            <artifactId>bom</artifactId>`
		`            <version>2.21.21[2]</version>`
		`            <type>pom</type>`
		`            <scope>import</scope>`
		`        </dependency>`
		`    </dependencies>`
		`</dependencyManagement>`
		`<dependencies>`
		`    <dependency>`
		`        <groupId>software.amazon.awssdk</groupId>`
		`        <artifactId>imds</artifactId>`
		`    </dependency>`
		`    <dependency>`

Change	v1	v2

		`<groupId>software.amazon.awssdk</groupId>`
		`<artifactId>apache-client [3]</artifactId>`
		`</dependency>`
		`</dependencies>`

Package name	`com.amazonaws.util`	`software.amazon.awssdk.imds`

Instantiation approach	Use static utility methods; no instantiation:
	`String localHostName = EC2MetadataUtils.getLocalHostName();`
		Use a static factory method:
		`Ec2MetadataClient client = Ec2MetadataClient.create();`
		Or use a builder approach:
		`Ec2MetadataClient client = Ec2MetadataClient.builder()`
		`.endpointMode(EndpointMode.IPV6)`
		`.build();`

Types of clients	Synchronous only utility methods: `EC2MetadataUtils`	Synchronous: `Ec2MetadataClient` Asynchronous: `Ec2MetadataAsyncClient`

[1] Latest version. [2] Latest version.

[3]Notice the declaration of the `apache-client` module for v2. V2 of the EC2 metadata utility requires an implementation of the `SdkHttpClient` interface for the synchronous metadata client,

or the `SdkAsyncHttpClient` interface for the asynchronous metadata client. The ??? section shows the list of HTTP clients that you can use.

Requesting metadata

In v1, you use static methods that accept no parameters to request metadata for an EC2 resource. In contrast, you need to specify the path to the EC2 resource as a parameter in v2. The following table shows the different approaches.

v1	v2

`String userMetaData = EC2MetadataUtils.getUserData();`	`Ec2MetadataClient client = Ec2MetadataClient.create();`
	`Ec2MetadataResponse response = client.get("/latest/user-data");`
	`String userMetaData = response.asString();`

Refer to the instance metadata categories to find the path you need to supply to request a piece of metadata.

Note

When you use an instance metadata client in v2, you should aim to use the same client for all request to retrieve metadata.

Behavior changes

JSON data

On EC2, the locally running Instance Metadata Service (IMDS) returns some metadata as JSON formatted strings. One such example is the dynamic metadata of an instance identity document.

The v1 API contains separate methods for each piece of instance identity metadata, whereas the v2 API directly returns the JSON string. To work with the JSON string, you can use the Document API to parse the response and navigate the JSON structure.

The following table compares how you retrieve metadata of an instance identity document in v1 and v2.

Use case	v1	v2
Retrieve the Region	`InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo(); String region = instanceInfo.getRegion();`	`Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String region = instanceInfo.asMap().get("region").asString();`
Retrieve the instance id	`InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo(); String instanceId = instanceInfo.instanceId;`	`Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceId = instanceInfo.asMap().get("instanceId").asString();`
Retrieve the instance type	`InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo(); String instanceType = instanceInfo.instanceType();`	`Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceType = instanceInfo.asMap`

Use case	v1	v2
		`().get("instanceType").asString();`

Endpoint resolution differences

The following table shows the locations that the SDK checks to resolve the endpoint to IMDS. The locations are listed in descending priority.

v1	v2
System property: `com.amazonaws.sdk.ec2MetadataServiceEndpointOverride`	Client builder configuration method: `endpoint(...)`
Environment variable: `AWS_EC2_METADATA_SERVICE_ENDPOINT`	System property: `aws.ec2MetadataServiceEndpoint`
Default Value: `http://169.254.169.254`	Config file: `~.aws/config` with the `ec2_metadata_service_endpoint` setting
	Value associated with resolved `endpoint-mode`
	Default value: `http://169.254.169.254`

Endpoint resolution in v2

When you explicitly set an endpoint by using the builder, that endpoint value takes priority over all other settings. When the following code executes, the `aws.ec2MetadataServiceEndpoint` system property and config file `ec2_metadata_service_endpoint` setting are ignored if they exist.

```
Ec2MetadataClient client = Ec2MetadataClient
  .builder()
  .endpoint(URI.create("endpoint.to.use"))
```

```
    .build();
```

Endpoint-mode

With v2, you can specify an endpoint-mode to configure the metadata client to use the default endpoint values for IPv4 or IPv6. Endpoint-mode is not available for v1. The default value used for IPv4 is `http://169.254.169.254` and `http://[fd00:ec2::254]` for IPv6.

The following table shows the different ways that you can set the endpoint mode in order of descending priority.

		Possible values
Client builder configuration method: `endpointMode(...)`	`Ec2MetadataClient client = Ec2MetadataClient.builder().endpointMode(EndpointMode.IPV4).build();`	`EndpointMode.IPV4` , `EndpointMode.IPV6`
System property	`aws.ec2MetadataServiceEndpointMode`	`IPv4`, `IPv6` (case does not matter)
Config file: `~.aws/config`	`ec2_metadata_service_endpoint` setting	`IPv4`, `IPv6` (case does not matter)
Not specified in the previous ways	IPv4 is used	

How the SDK resolves `endpoint` or `endpoint-mode` in v2

1. The SDK uses the value that you set in code on the client builder and ignores any external settings. Because the SDK throws an exception if both `endpoint` and `endpointMode` are called on the client builder, the SDK uses the endpoint value from whichever method you use.
2. If you do not set a value in code, the SDK looks to external configuration—first for system properties and then for a setting in the config file.
    a. The SDK first checks for an endpoint value. If a value is found, it is used.

b. If the SDK still hasn't found a value, the SDK looks for endpoint mode settings.

3. Finally, if the SDK finds no external settings and you have not configured the metadata client in code, the SDK uses the IPv4 value of `http://169.254.169.254`.

IMDSv2

Amazon EC2 defines two approaches to access instance metadata:

- Instance Metadata Service Version 1 (IMDSv1) – Request/response approach
- Instance Metadata Service Version 2 (IMDSv2) – Session-oriented approach

The following table compares how the Java SDKs work with IMDS.

v1	v2
IMDSv2 is used by default	Always uses IMDSv2
Attempts to fetch a session token for each request and falls back to IMDSv1 if it fails to fetch a session token	Keeps a session token in an internal cache that is reused for multiple requests

The SDK for Java 2.x supports only IMDSv2 and does not fall back to IMDSv1.

Configuration differences

The following table lists the differing configuration options.

Configuration	v1	v2
Retries	Configuration not available	Configurable through builder method `retryPolicy(...)`
HTTP	Connection timeout configurable through the `AWS_METADATA_SERVICE_TIMEOU`	Configuration available by passing an HTTP client to the builder method `httpClient(...)`. The default

Configuration	v1	v2
	T environment variable. The default is 1 second.	connection timeout for HTTP clients is 2 seconds.

Example v2 HTTP configuration

The following example shows how you can configure the metadata client. This example configures the connection timeout and uses the Apache HTTP client.

```
SdkHttpClient httpClient = ApacheHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(1))
    .build();

Ec2MetadataClient imdsClient = Ec2MetadataClient.builder()
    .httpClient(httpClient)
    .build();
```

Changes in Amazon CloudFront presigning from version 1 to version 2

This topic details the changes in the Amazon CloudFront from version 1 (v1) to version 2 (v2).

High-level changes

Change	v1	v2

Maven dependencies	`<dependencyManagement>`
	`<dependencies>`
	`<dependency>`
	`<groupId>com.amazonaws</groupId>`
	`<artifactId>aws-java-sdk-bom</artifactId>`
	`<version>1.12.587[1]</version>`
	`<type>pom</type>`
		`<dependencyManagement>`
		`<dependencies>`
		`<dependency>`
		`<groupId>software.amazon.awssdk</groupId>`
		`<artifactId>bom</artifactId>`
		`<version>2.21.21[2]</version>`
		`<type>pom</type>`
		`<scope>import</scope>`

Change	v1	v2

	`<scope>import</scope>`
	`</dependency>`
	`</dependencies>`
	`</dependencyManagement>`
	`<dependencies>`
	`<dependency>`
	`<groupId>com.amazonaws</groupId>`
	`<artifactId>cloudfront</artifactId>`
	`</dependency>`
	`</dependencies>`
		`</dependency>`
		`</dependencies>`
		`</dependencyManagement>`
		`<dependencies>`
		`<dependency>`
		`<groupId>software.amazon.awssdk</groupId>`
		`<artifactId>cloudfront</artifactId>`
		`</dependency>`
		`</dependencies>`

Package name	`com.amazonaws.services.cloudfront`	`software.amazon.awssdk.services.cloudfront`
Class names	CloudFrontUrlSigner CloudFrontCookieSigner	CloudFrontUtilities SignedUrl CannedSignerRequest CustomSignerRequest

[1] Latest version. [2] Latest version.

API changes

Behavior	v1	v2
Build a canned request	Arguments are passed directly to the API.	`CannedSignerRequest cannedRequest =` `CannedSignerRequest.builder()`

Behavior	v1	v2
		`.resourceUrl(resourceUrl)` `.privateKey(privateKey)` `.keyPairId(keyPairId)` `.expirationDate(expirationDate)` `.build();`
Build a custom request	Arguments are passed directly to the API.	`CustomSignerRequest customRequest = CustomSignerRequest.builder()` `.resourceUrl(resourceUrl)` `.privateKey(keyFile)` `.keyPairId(keyPairId)` `.expirationDate(expirationDate)` `.activeDate(activeDate)` `.ipRange(ipRange)` `.build();`

<table>
<tr><th>Behavior</th><th>v1</th><th>v2</th></tr>
<tr><td>Generate a signed URL (canned)</td><td><pre><code>String signedUrl =
  CloudFrontUrlSigne
r.getSignedURLWith
CannedPolicy(
    resourceUrl,
 keyPairId, privateKe
y, expirationDate);</code></pre></td><td><pre><code>CloudFrontUtilities
 cloudFrontUtilities =
    CloudFrontUtilitie
s.create();

SignedUrl signedUrl =

    cloudFrontUtilitie
s.getSignedUrlWith
CannedPolicy(canne
dRequest);

String url = signedUrl
.url();</code></pre></td></tr>
<tr><td>Generate a signed cookie (custom)</td><td><pre><code>CookiesForCustomPolicy
 cookies =
    CloudFrontCookieSi
gner.getCookiesFor
CustomPolicy(
        resourceUrl,
 privateKey, keyPairId
, expirationDate,
        activeDate,
 ipRange);</code></pre></td><td><pre><code>CloudFrontUtilities
 cloudFrontUtilities =
    CloudFrontUtilitie
s.create();

CookiesForCustomPolicy
 cookies =
     cloudFrontUtilitie
s.getCookiesForCus
tomPolicy(customRe
quest);</code></pre></td></tr>
</table>

Refactored cookie headers in v2

In Java v1, the Java SDK delivers cookie headers as a `Map.Entry<String, String>`.

```
Map.Entry<String, String> signatureMap = cookies.getSignature();
String signatureKey = signatureMap.getKey(); // "CloudFront-Signature"
String signatureValue = signatureMap.getValue(); // "[SIGNATURE_VALUE]"
```

The Java v2 SDK delivers the entire header as a single `String`.

```
String signatureHeaderValue = cookies.signatureHeaderValue(); // "CloudFront-Signature=[SIGNATURE_VALUE]"
```

Changes in parsing Amazon S3 URIs from version 1 to version 2

This topic details the changes in parsing Amazon S3 URIs from version 1 (v1) to version 2 (v2.).

High-level changes

To begin parsing an S3 URI in v1, you instantiate an `AmazonS3URI` by using a constructor. In v2 you call `parseUri()` on an instance of `S3Utilities`, to return an `S3URI`.

Change	v1	v2

Maven dependencies	`<dependencyManagement>`
	`<dependencies>`
	`<dependency>`
	`<groupId>com.amazonaws</groupId>`
	`<artifactId>aws-java-sdk-bom</artifactId>`
	`<version>1.12.587</version>` [1]
	`<type>pom</type>`
	`<scope>import</scope>`
	`</dependency>`
	`</dependencies>`
	`</dependencyManagement>`
	`<dependencies>`
	`<dependency>`
	`<groupId>com.amazonaws</groupId>`
	`<artifactId>s3</artifactId>`
	`</dependency>`
		`<dependencyManagement>`
		`<dependencies>`
		`<dependency>`
		`<groupId>software.amazon.awssdk</groupId>`
		`<artifactId>bom</artifactId>`
		`<version>2.21.21</version>` [2]
		`<type>pom</type>`
		`<scope>import</scope>`
		`</dependency>`
		`</dependencies>`
		`</dependencyManagement>`
		`<dependencies>`
		`<dependency>`
		`<groupId>software.amazon.awssdk</groupId>`
		`<artifactId>s3</artifactId>`
		`</dependency>`

Change	v1	v2
	`</dependencies>`	`</dependencies>`
Package name	`com.amazonaws.services.s3`	`software.amazon.awssdk.services.s3`
Class names	AmazonS3URI	S3URI

[1] Latest version. [2] Latest version.

API changes

Behavior	v1	v2

Parse an S3 URI.	`URI uri = URI.create( "https://s3.amazonaws.com");`
	
	`AmazonS3Uri s3Uri =`
	`    new AmazonS3URI(uri, false);`
		`S3Client s3Client = S3Client.create();`
		`S3Utilities s3Utilities =`
		`    s3Client.utilities();`
		
		`S3Uri s3Uri =`
		`    s3Utilities.parseUri(uri);`

Retrieve the bucket name from an S3 URI.	`String bucket = s3Uri.getBucket();`	`Optional<String> bucket = s3Uri.bucket();`
Retrieve the key.	`String key = s3Uri.getKey();`	`Optional<String> key = s3Uri.key();`

Retrieve the region.	`String region = s3Uri.getRegion();`	`Optional<Region> region = s3Uri.region();`
		
		`String region;`
		`if (s3Uri.region().isPresent()) {`

Behavior	v1	v2
		`region = s3Uri.region().get().id();` `}`
Retrieve whether the S3 URI is path style.	`boolean isPathStyle = s3Uri.isPathStyle();`	`boolean isPathStyle = s3Uri.isPathStyle();`
Retrieve the version ID.	`String versionId = s3Uri.getVersionId();`	`Optional<String> versionId = s3Uri.firstMatchingRawQueryParameter("versionId");`
Retrieve the query parameters.	N/A	`Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();`

Behavior changes

URL encoding

v1 provides the option to pass in a flag to specify whether the URI should be URL encoded. The default value is `true`.

In v2, URL encoding is not supported. If you work with object keys or query parameters that have reserved or unsafe characters, you must URL encode them. For example you need to replace a whitespace `" "` with `%20`.

Changes in the IAM Policy Builder API from version 1 to version 2

This topic details the changes in the IAM Policy Builder API from version 1 (v1) to version 2 (v2).

High-level changes

Change	v1	v2
Maven dependencies	<dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</groupId> <artifactId>aws-java-sdk-bom</artifactId> <version>1.12.587[1]</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</groupId> <artifactId>aws-java-sdk-core</artifactId> </dependency> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>bom</artifactId> <version>2.21.21[2]</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>iam-policy-builder</artifactId> </dependency> </dependencies>
Package name	`com.amazonaws.auth.policy`	`software.amazon.awssdk.policybuilder.iam`
Class names	Policy Statement	IamPolicy IamStatement

Change	v1	v2
	• Statement.Effect • IdentiyManagementActions • Resource • Principal • Condition	• IamEffect • IamAction • IamResource • IamPrincipal • IamCondition • IamConditionOperator • IamConditionKey

[1] Latest version. [2] Latest version.

API changes

Setting	v1	v2
Instantiate a policy	`Policy policy = new Policy();`	`IamPolicy.Builder policyBuilder = IamPolicy.builder(); ... IamPolicy policy = policyBuilder.build();`
Set id	`policy.withtId(...);` `policy.setId(...);`	`policyBuilder.id(...);`
Set version	N/A - uses default version of 2012-10-17	`policyBuilder.version(...);`
Create statement	`Statement statement = new Statement(Effect.Allow) .withActions(...)`	`IamStatement statement = IamStatement.builder() .effect(IamEffect.ALLOW)`

Setting	v1	v2

	`.withConditions(...)`
	`.withId(...)`
	`.withPrincipals(...)`
	`.withResources(...);`
		`.actions(...)`
		`.notActions(...)`
		`.conditions(...)`
		`.sid(...)`
		`.principals(...)`
		`.notPrincipals(...)`
		`.resources(...)`
		`.notResources(...)`
		`.build()`

Set statement	`policy.withStatements(statement);` `policy.setStatements(statement);`	`policyBuilder.addStatement(statement);`

Differences in building a statement

Actions

v1

The v1 SDK has enum types for service actions that represent *`Action`* elements in a policy statement. The following `enum` types are some examples.

- `IdentityManagementActions`
- `DynamoDBv2Actions`
- `SQSActions`

The following example shows the `SendMessage` constant for `SQSActions`.

```
Action action = SQSActions.SendMessage;
```

You cannot specify a *`NotAction`* element to a statement in v1.

v2

In v2, the IamAction interface represents all actions. To specify a service-specific action element, pass a string to the `create` method as shown in the following code.

```
IamAction action = IamAction.create("sqs:SendMessage");
```

You can specify a *`NotAction`* for a statement with v2 as shown in the following code.

```
IamAction action = IamAction.create("sqs:SendMessage");
IamStatement.builder().addNotAction(action);
```

Conditions

v1

To represent statement conditions, the v1 SDK uses subclasses of `Condition`.

- ArnCondition
- BooleanCondition
- DateCondition
- IpAddressCondition
- NumericCondition
- StringCondition

Each `Condition` subclass defines a comparison `enum` type to help define the condition. For example, the following shows a *not like* string comparison for a condition.

```
Condition condition = new StringCondition(StringComparisonType.StringNotLike, "key",
 "value");
```

v2

In v2, you build a condition for a policy statement by using `IamCondition` and provide an `IamConditionOperator`, which contains `enums` for all types.

```
IamCondition condition = IamCondition.create(IamConditionOperator.STRING_NOT_LIKE,
 "key", "value");
```

Resources

v1

A policy statement's *Resource* element is represented by the SDK's Resource class. You supply the ARN as a string in the constructor. The following subclasses provide convenience constructors.

- S3BucketResource
- S3ObjectResource
- SQSQueueResource

In v1, you can specify a *NotResource* element for a Resource by calling the withIsNotType method as shown in the following statement.

```
Resource resource = new Resource("arn:aws:s3:::mybucket").withIsNotType(true);
```

v2

In v2, you create a *Resource* element by passing an ARN to the IamResource.create method.

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");
```

An IamResource can be set as *NotResource* element as shown in the following snippet.

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");
IamStatement.builder().addNotResource(resource);
```

IamResource.ALL represents all resources.

Principals

v1

The v1 SDK offers the following Principal classes to represent types of principals that include all members:

- `AllUsers`
- `AllServices`
- `AllWebProviders`
- `All`

You cannot add a *`NotPrincipal`* element to a statement.

v2

In v2, `IamPrincipal.ALL` represents all principals:

To represent all members in other types of principals, use the `IamPrincipalType` classes when you create a `IamPrincipal`.

- `IamPrincipal.create(IamPrincipalType.AWS,"*")` for all users.
- `IamPrincipal.create(IamPrincipalType.SERVICE,"*")` for all services.
- `IamPrincipal.create(IamPrincipalType.FEDERATED,"*")` for all web providers.
- `IamPrincipal.create(IamPrincipalType.CANONICAL_USER,"*")` for all canonical users.

You can use the `addNotPrincipal` method to represent a *`NotPrincipal`* element when you create a policy statement as shown in the following statement.

```
IamPrincipal principal = IamPrincipal.create(IamPrincipalType.AWS,
 "arn:aws:iam::444455556666:root");
IamStatement.builder().addNotPrincipal(principal);
```

Use the SDK for Java 1.x and 2.x side-by-side

You can use both versions of the AWS SDK for Java in your projects.

The following shows an example of the `pom.xml` file for a project that uses Amazon S3 from version 1.x and DynamoDB from version 2.16.1.

Example Example of POM

This example shows a `pom.xml` file entry for a project that uses both 1.x and 2.x versions of the SDK.

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-bom</artifactId>
            <version>1.12.1</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <dependency>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>bom</artifactId>
          <version>2.16.1</version>
          <type>pom</type>
          <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-s3</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb</artifactId>
    </dependency>
</dependencies>
```

OpenPGP key for the AWS SDK for Java

All publicly available Maven artifacts for the AWS SDK for Java are signed using the OpenPGP standard. The public key that you need to verify the signature of an artifact is available in the following section.

Current key

The following table shows OpenPGP key information for the current releases of the SDK for Java 1x and SDK for Java 2.x.

Key ID	0xAC107B386692DADD
Type	RSA
Size	4096/4096
Created	2016-06-30
Expires	2024-10-08
User ID	AWS SDKs and Tools <aws-dr-tools@amazon.com>
Key fingerprint	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

To copy the following OpenPGP public key for the SDK for Java to the clipboard, select the "Copy" icon in the upper right corner.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

xsFNBFd1gAUBEACqbmmFbxdJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYpOviSWsX2psgvdmeyUpW9ap0lrThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKROmRODzUuaokLPo24pfm9bnr1RnRtwt5ktPAA5bM9ZZaGKriej
kT2lPffbBjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WPlnxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9bOie
```

```
nj3QxLuQ1ZUKF79ES6JaM4tOz1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+XfOCl6byOJFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaSOSZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVBOHKrq+fluorWepQivctzN6Y1NOkx5naTPGGaKWK7G2q
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNArPWb3dPMThL2xAY+fS60vXdB1SkOtYJpDWpFgvo0d+VQ+
hV6XulGAHAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/O6Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDEOO62kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SJQ8jl+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJKUOb6b1786WNySIzF2gxqlkkEmBl4RAssQkeXjrSmGws
MDyHNqyJeYFusl8sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MYlnasd4bW2RK1sr7plkBf8QRe6biiQRF3KD
OSn5CbmXpAcHJ1ZHzRRdkXZDNQC6vCJxsy13O0TrhJtAV1Yq347uyUbVi291ISVg
roUVtprsmHoEk5GoOTHbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs
/Hd981FdVghYYvq//gTAkJk0WxfGqO30wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsFlBBgBCgAPBQJXdYAFAhsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0HOVWHwQsaW
jMrGj0OOMFzxGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxpyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+YnOQf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
l6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJBlkke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeql67ggx+wFjKDcfhZA7m7OLHODysrGVCLcmuinUBaNlHmLDcGY
XZ+kMCoXfObpuCVByQmNJgEb47EIFlx/+TEeNHKMO+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv0OPCvO
WTA/MClv28y0PrSvcvMXnduLtkBEX7TISMPW+n+OTa63/z4YFfEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```

Document history

This topic describes important changes to the AWS SDK for Java Developer Guide over the course of its history.

This guide was last published on **June 18, 2024**.

Change	Description	Date
the section called “Retries”	Add retry strategy topic	June 18, 2024
How to set the JVM TTL	Remove instructions to set `networkaddress.cache.ttl` security property by using a java command-line system property.	May 21, 2024
the section called “Reduce SDK startup time for AWS Lambda”	Update HTTP client recommendation to reduce startup time for AWS Lambda	May 14, 2024
the section called “Service client metrics”	Reorganize metrics table items	May 1, 2024
the section called “Troubleshooting”	Add troubleshooting topic.	April 26, 2024
the section called “Metrics collected with each request”	Add new metrics reported by the SDK.	April 26, 2024
the section called “Set the JVM TTL for DNS name lookups”	Change recommended DNS lookup TTL to 5 seconds.	April 23, 2024
the section called “Package name to artifactId mappings”	Add package name to Maven artifactId mapping topic.	April 17, 2024
the section called “Use SDK metrics”	Add configuration details to the metrics section.	April 12, 2024

Change	Description	Date
the section called "IAM Policy Builder API"	Add IAM Policy Builder API migration information.	April 11, 2024
???	Update HTTP proxy informati on.	April 3, 2024
the section called "Securely"	Add instructions to disable IMDSv1.	March 14, 2024
the section called "Step-by-step instructions"	Add step-by-step migration instructions.	March 8, 2024
Migrate to version 2	Update migration topic.	February, 14, 2024
the section called "Configure AWS CRT-based HTTP clients"	Add information about the synchronous AWS CRT-based HTTP client.	January 5, 2024
the section called "Amazon Cognito Identity" and the section called "Amazon Cognito Identity Provider"	Amazon Cognito examples moved to Code Examples section.	December 28, 2023
Use SDK features	Reworked the SDK features topic.	December 11, 2023
OpenPGP key	Provide current OpenPGP key.	December 6, 2023
the section called "Serializ ation changes"	Describe serialization differences between v1 and v2 of the SDK for Java.	December 5, 2023
the section called "S3 Transfer Manager"	Add a section that details the changes in the S3 Transfer Manager from version 1 to version 2.	November 13, 2023

Change	Description	Date
the section called “Annotation reference”	Add a listing of data class annotations that can be used with the DynamoDB Enhanced Client.	October 30, 2023
???	Add information on the migration status of libraries and utilities from SDK for Java v1.x to v2.x	October 17, 2023
???	Update the Gradle setup topic	October 17, 2023
the section called “Ignore null attributes of nested objects”	Add information about the DynamoDB Enhanced Client `@DynamoDbIgnoreNulls` annotation.	September 22, 2023
the section called “Cross-Region access”	Add information about cross-Region access to Amazon S3 buckets.	August 31, 2023
the section called “Preserve empty objects”	Add section that discusses the `@DynamoDbPreserveEmptyObject` annotation.	August 25, 2023
???	Update service client section.	August 15, 2023
the section called “Client recommendations”	Since version 0.23, AWS CRT supports musl-based OS such as Alpine Linux. HTTP client recommendations now reflect the musl support.	August 11, 2023
the section called “Create IAM policies”	Add IAM Policy Builder API section	July 31, 2023

Change	Description	Date
the section called "Get started"	Correct several snippets in the Get Started section of the DynamoDB Enhanced Client topic.	July 24, 2023
the section called "Configure HTTP proxies"	Add HTTP proxy support information and examples for each HTTP client.	June 2, 2023
Reorganize the table of contents	Promote *Code examples* section and *Work with AWS services* to top-level TOC entries.	May 24, 2023
the section called "Add logging dependency"	Show Gradle dependencies in logging section.	May 23, 2023
the section called "Work with paginated results"	Update pagination topic.	May 18, 2023
the section called "Set up a Gradle project"	Update Gradle project setup.	May 3, 2023
DynamoDB Enhanced Client API	Rewritten DynamoDB Enhanced Client API topic released.	April 28, 2023
Update get started tutorial instructions	Maven archetype modified to include option for credentia lsProvider; instructions modified accordingly.	April 11, 2023
the section called "Client recommendations"	Add HTTP client decision guidance	March 30, 2023

Change	Description	Date
IAM best practices updates	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM.	March 14, 2023
the section called “Reload profile credentials”	Add section on reloading profile credentials.	February 9, 2023
the section called “Configure AWS CRT-based HTTP clients”	Update topic for GA release.	February 8, 2023
the section called “Work with Amazon EC2 instance metadata”	Add guided example for Java SDK client for Amazon S3 instance metadata service.	February 1, 2023
the section called “Use a performant S3 client”	Add section for the AWS CRT-based S3 Client.	December 19, 2022
the section called “Transfer files and directories”	Update Amazon S3 Transfer Manager examples for GA release.	December 19, 2022
the section called “Best practices”	Added best practices section.	November 18, 2022
the section called “Load temporary credentials from an external process”	Added section on loading credentials from an external process.	November 15, 2022
the section called “Service client metrics”	Updated metric listing with HTTP client usage requireme nt.	November 9, 2022
the section called “Transfer files and directories”	Example code corrected.	November 2, 2022

Change	Description	Date
the section called “Reduce SDK startup time for AWS Lambda”	Updated section with additional options to reduce Lambda startup time.	November 1, 2022
the section called “HTTP clients”	Added configuration information to cover all HTTP clients in the SDK.	October 26, 2022
the section called “Logging”	Updated logging topic to include wire logging details for all HTTP clients.	October 4, 2022
the section called “AWS database services”	Added overview section of AWS database services and the SDK for Java 2.x.	September 13, 2022
EC2-Classic Networking is Retiring	EC2-Classic is retiring on August 15, 2022.	July 28, 2022
the section called “Additional authentication options”	Update to dependency required for single sign-on authentication.	July 18, 2022
the section called “Transport Layer Security (TLS)”	Update TLS security informati on.	April 8, 2022
the section called “Additional authentication options”	Added more information about setting up and using credentials.	February 22, 2021
the section called “Set up a GraalVM Native Image project”	New topic for setting up a GraalVM Native Image project.	February 18, 2021
the section called “Poll for resource states”	Waiters released; added topic for the new feature.	September 30, 2020

Change	Description	Date
the section called “Use SDK metrics”	Metrics released; added topic for the new feature.	August 17, 2020
the section called “Amazon SNS”	Added example topics for Amazon SNS.	May 30, 2020
the section called “Reduce SDK startup time for AWS Lambda”	Added AWS Lambda function performance topic.	May 29, 2020
the section called “Set the JVM TTL for DNS name lookups”	Added JVM TTL DNS caching topic.	April 27, 2020
the section called “Set up an Apache Maven project”, the section called “Set up a Gradle project”	New Maven and Gradle set up topics.	April 21, 2020
the section called “Transport Layer Security (TLS)”	Added TLS 1.2 to security section.	March 19, 2020
the section called “Subscribe to Amazon Kinesis Data Streams”	Added Kinesis stream examples.	August 2, 2018
the section called “Work with paginated results”	Added auto pagination topic.	April 5, 2018
???	Added example topics for IAM, Amazon EC2, CloudWatch and DynamoDB.	December 29, 2017
the section called “Amazon S3”	Added getobjects example for Amazon S3.	August 7, 2017

Change	Description	Date
the section called “Use asynchronous programming”	Added async topic.	August 4, 2017
GA release of the AWS SDK for Java 2.x	AWS SDK for Java version 2 (v2) released.	June 28, 2017